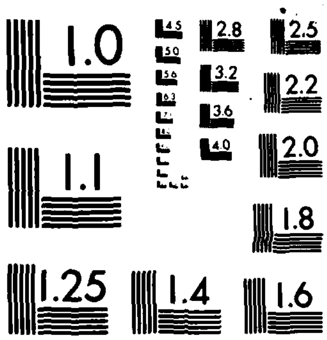
AD-A141 309 ADAPTIVE TOPOLOGICAL CONFIGURATION OF AN INTEGRATED CIRCUIT/PACKET-SWITCHED COMPUTER NETWORK(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH M J KIEMELE 1984 1/3

UNCLASSIFIED AFIT/CI/NR-84-16D F/G 17/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 84-16D	2. GOVT ACCESSION NO. AD-A141309	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Adaptive Topological Configuration Of An Integrated Circuit/Packet-Switched Computer Network		5. TYPE OF REPORT & PERIOD COVERED ~~THESIS~~/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mark Jay Kiemele		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Texas A&M University		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE 1984
		13. NUMBER OF PAGES 219
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC ELECTE MAY 22 1984 S D A

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

Lynn E. Wolaver 15 May 84

LYNN E. WOLAVER
Dean for Research and Professional Development
AFIT, Wright-Patterson AFB OH

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

84 05 21 211

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A141 309

DTIC FILE COPY

ABSTRACT

Adaptive Topological Configuration of an Integrated Circuit/Packet-Switched Computer Network. (May 1984)

Mark Jay Kiemele, B.S., North Dakota State University; M.S., North Dakota State University

Chairman of Advisory Committee: Dr. Udo W. Pooch

This research investigates the performance and design of integrated circuit/packet-switched computer-communication networks. The integrated network under consideration has a circuit-switched communications subnet whose trunk lines carry both voice and data simultaneously. Peripheral packet switches provide data subscribers access tc the subnet, while voice subscribers terminate directly to the circuit switch nodes of the subnet.

An existing simulation model is modified and subsequently used to analyze the performance of integrated networks. A simulation experiment is designed and conducted to identify key network parameters and to determine the relationships that exist between these parameters and network performance. Multiple regression analyses are conducted to obtain models that describe the mean behavior of certain network performance measures with respect to changes in network traffic load, link capacity,

and network size.

The exact solution to the integrated network topology design problem is seen to be intractable for even small networks. This dissertation addresses the topology design problem using an iterative, heuristic approach whereby many suboptimal solutions (local minima) are generated in lieu of one optimal solution. The iterative scheme integrates the simulator as a network performance generation device into a performance feedback loop which dynamically reconfigures the network topology. An integrated cut-saturation add heuristic and a reliability-preserving delete heuristic are developed to move the network topology in the direction of an optimal, feasible solution.

The topology design methodology is implemented as a modularized FORTRAN program and is applied to several networks of varying design specifications. Results demonstrate that the methodology developed constitutes a viable tool that can be used to analyze, design, and modify the integrated networks of the future.

Author: MARK JAY KIEMELE
MAJOR, USAF
1984

Title: ADAPTIVE TOPOLOGICAL CONFIGURATION OF AN INTEGRATED CIRCUIT/PACKET-SWITCHED COMPUTER NETWORK

Number of Pages: 219 (includes 14 introductory pages)

Degree Awarded: Ph. D., Computer Science
Texas A&M University

AFIT/CI/NR 84-16D

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (ATC). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: Adaptive Topological Configuration of an Integrated Circuit/Packet-Switched Computer Network

AUTHOR: Mark Jay Kiemele

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS ________ ( ) b. $________

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY SIGNIFICANT ( ) b. SIGNIFICANT ( ) c. SLIGHTLY SIGNIFICANT ( ) d. OF NO SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME GRADE POSITION

ORGANIZATION LOCATION

STATEMENT(s):

ADAPTIVE TOPOLOGICAL CONFIGURATION OF AN INTEGRATED CIRCUIT/PACKET-SWITCHED COMPUTER NETWORK

A Dissertation

by

MARK JAY KIEMELE

Submitted to the Graduate College of
Texas A&M University
in partial fulfillment of the requirements for the degree
of

DOCTOR OF PHILOSOPHY

May 1984

Major Subject: Computer Science

ADAPTIVE TOPOLOGICAL CONFIGURATION OF AN INTEGRATED CIRCUIT/PACKET-SWITCHED COMPUTER NETWORK

A Dissertation

by

MARK JAY KIEMELE

Approved as to style and content by:

Udo W. Pooch
(Chairman of Committee)

Sallie V. Sheppard
(Member)

Robert L. Sielken, Jr.
(Member)

Glen N. Williams
(Member)

Bruce H. McCormick
(Head of Department)

May 1984

ABSTRACT

Adaptive Topological Configuration of an Integrated Circuit/Packet-Switched Computer Network. (May 1984)

Mark Jay Kiemele, B.S., North Dakota State University; M.S., North Dakota State University

Chairman of Advisory Committee: Dr. Udo W. Pooch

This research investigates the performance and design of integrated circuit/packet-switched computer-communication networks. The integrated network under consideration has a circuit-switched communications subnet whose trunk lines carry both voice and data simultaneously. Peripheral packet switches provide data subscribers access to the subnet, while voice subscribers terminate directly to the circuit switch nodes of the subnet.

An existing simulation model is modified and subsequently used to analyze the performance of integrated networks. A simulation experiment is designed and conducted to identify key network parameters and to determine the relationships that exist between these parameters and network performance. Multiple regression analyses are conducted to obtain models that describe the mean behavior of certain network performance measures with respect to changes in network traffic load, link capacity,

and network size.

The exact solution to the integrated network topology design problem is seen to be intractable for even small networks. This dissertation addresses the topology design problem using an iterative, heuristic approach whereby many suboptimal solutions (local minima) are generated in lieu of one optimal solution. The iterative scheme integrates the simulator as a network performance generation device into a performance feedback loop which dynamically reconfigures the network topology. An integrated cut-saturation add heuristic and a reliability-preserving delete heuristic are developed to move the network topology in the direction of an optimal, feasible solution.

The topology design methodology is implemented as a modularized FORTRAN program and is applied to several networks of varying design specifications. Results demonstrate that the methodology developed constitutes a viable tool that can be used to analyze, design, and modify the integrated networks of the future.

ACKNOWLEDGEMENTS

The author's graduate studies at Texas A&M University were funded by the United States Air Force through a program administered by the Air Force Institute of Technology (AFIT), Wright-Patterson AFB, Ohio.

I extend my deepest gratitude to Dr. Udo W. Pooch, my chairman and friend, for his professional and personal guidance throughout my tenure at Texas A&M University. Without his professional expertise, direction, and incredible ability to motivate, successful completion of this research would not have been possible.

Sincere appreciation is extended to Dr. Sallie V. Sheppard and Dr. Glen N. Williams, Computer Science, Dr. Robert L. Sielken, Jr., Statistics, and Dr. Worth D. Nowlin, Jr. for their technical support, suggestions, and encouragement. Special thanks is also due to Drs. R. J. Freund and R. R. Hocking, Statistics, for their assistance.

Recognition is extended to Debbie Callaway whose accurate and timely processing of this dissertation is sincerely appreciated.

A very special note of gratitude is extended to my wife, Carol, who had to assume all of the parental duties at home and whose patience and perseverance were endless; and to my son, Kyle, who did not understand but tried.

To my wife, Carol, who believes
these are the good old days.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

CHAPTER I

INTRODUCTION

1. Computer-Communication Networks

The explosive evolution of the computer and communications technologies has resulted in a marriage which is rapidly forming the basis of our information society. A computer-communication network may be described as an interconnected group of independent computer systems which communicate with one another [12]. The reasons for communicating are varied. They may include the sharing of resources, such as programs, data, hardware, or software; or it may be that a computer system is used solely as a communications processor for information transfer and/or controlling terminals [79].

Sometimes a subtle distinction is made between a computer-communication network and a computer network. The difference between the two, according to Elovitz and Heitmeyer [26], is that in a 'computer-communication network', the user is responsible for managing the computer resources; while in a 'computer network', the

---

The Communications of the Association for Computing Machinery is used as a pattern for format and style.

resources are managed automatically by a network operating system. In this research, the two terms will not be distinguished and will be used interchangeably, along with the term 'network'.

The emphasis of this research is on the topology of a network. Hence, the graph-theoretic aspects of a network are of major importance. In this regard, a computer network consists of a communications subnet (or backbone) together with the facilities needed to gain access to the subnet. The backbone of the network is comprised of communications processors (nodes) and trunk lines (links) which interconnect the nodes. The terms 'vertices' and 'nodes' are synonymous, and the terms 'link', 'arc', and 'edge' all may be used to denote a connector between two vertices in a graph or a transmission medium connecting two nodes in a network.

1.1 Historical Development

Communications and computers make an unlikely partnership in the sense that communication is as old as man and computers have just arrived on the scene. The merging of the two technologies can be traced back to approximately 1968. At that time computer networks took the form of time-shared systems in which sharing of CPU time was the main rationale for the network [1]. At about the same time, the Carterfone Decision [61] gave both

common carriers and non-common carriers the legal right to expand the types of communication services that could be offered to their customers [19, 37, 54, 63, 100]. This act, together with improved transmission technology, led to computer users sharing not only CPU time, but the resources available at the computing center as well. This included peripheral devices, memory, and software. The goal of increased resource sharing spawned numerous computer networks [4, 28].

One of these is the Advanced Research Projects Agency network (ARPANET) which is generally considered the pioneer of all computer networks [1, 3, 22, 92]. The original goals of ARPANET were to allow computer resource sharing among several Department of Defense (DOD) research centers and to provide a live research environment for exploring the technical problems involved in networking. ARPANET, once an experimental network, has expanded from a 4-node network in early 1970 to more than 250 nodes today. No longer an experimental network, ARPANET allows the sharing of distributed data bases and computing resources among thousands of users at universities and research agencies across the United States and Europe.

The success of ARPANET is in good part due to a newer method of transmission, packet switching. A packet-switched system is one in which the transmitted message is segmented into smaller fixed size blocks or

packets, each having its own copy of the destination attached. These packets traverse the network independently until they reach the destination node, where they are reassembled into the original message. ARPANET has demonstrated that packet switching is much more efficient for the transmission of data than circuit switching. Circuit switching, on the other hand, is an older technique that was designed originally for voice transmission but has also been used for data transmission. In a circuit-switched network, a complete circuit (communications path) is established between the source and destination nodes prior to the start of communication, and all information flowing between these two nodes traverses the same path.

The proliferation of computer networks in the last decade is evidence that computer networking has proven itself in part as a cost-effective tool for communication activities as well as computer resource sharing. Comprehensive treatments of the classification and descriptions of existing computer-communication networks are abundant in the literature [1, 3, 4, 22, 28, 74, 92, 95].

1.2 Motivating an Integrated Circuit/Packet-Switched Network

Current military communication systems are generally designed to handle either voice calls or data transactions

but not both. Such deployed systems use separate facilities for the two classes of traffic, thereby magnifying both the manpower and maintenance problems that already exist. The grade of service for these systems is usually satisfactory, but crisis situations can and do force traffic flows that exceed system capabilities [5]. Recent Defense Communication Agency (DCA) studies have shown DOD's intent to implement an all-digital, integrated network that would be operational early in the next decade [5, 16, 80, 88]. Such a network would transmit voice and multiple classes of data (e.g., interactive, bulk, facsimile) simultaneously on a common transmission medium. The feasibility of such networks has been demonstrated by Dysart et al. [25] who contend that "the future for fully digital integrated voice and data transmission is very promising". The concept of integrating voice and data rests on the fact that speech can be digitized and thus can be handled under packet switching schemes.

Other recent studies have also addressed the problem of transmitting voice and data in the same computer-communication network [17, 29, 30, 33, 46, 51, 62, 76, 81, 82, 83, 84, 90, 94]. Gruber [50] aptly summarizes these studies by stating that "the motivations for considering mixed voice and data traffic...include: the advent of new voice related applications with the technology now existing to economically support them,

and...economy and flexibility. Perhaps the ultimate objectives of integration are...to realize the economics of equipment commonality, large-scale integration, higher resource utilization, and combined network operations, maintenance, and administrative policies".

The scenario to accomplish such integration has also been investigated [5, 16, 18, 25, 33, 53, 55, 69, 71, 83, 84, 94, 96, 97]. Despite the many tradeoffs between packet switching and circuit switching, the consensus is that circuit switching delays have been improved to the point where both circuit switching and packet switching can be employed advantageously in the same network [8]. It is this approach that is examined in this research.

1.3 Research Objectives and Plans

This dissertation investigates the performance and design of integrated circuit/packet-switched computer-communication networks. Specifically, the major goals of this research are to:

(1) Obtain and analyze performance data from such integrated networks. A computer network simulation model [15], or simulator, which simulates an integrated computer network, will be used to obtain the performance data.

(2) Identify key network parameters and determine the relationships that exist between these

parameters and the network's topology and performance.

(3) Develop a methodology which is designed to dynamically reconfigure the network topology in an attempt to satisfy specified performance constraints at minimum cost.

(4) Demonstrate the applicability of the methodology to the topological design of integrated networks.

This research will culminate in the delivery of a methodology which can be used by researchers and computer network designers alike. The model or tool to be produced will be flexible in the sense that designers could use it either to design new computer networks or to modify existing networks. Additionally, this research should produce further avenues of research whereby the performance of integrated networks may be enhanced.

1.4 Overview

Chapter II presents a review of the pertinent literature. Its purpose is to highlight the work that has been done in the design and analysis of integrated networks and to provide some background for this research.

The tool used to generate network performance data is described in Chapter III. The integration and queueing concepts implemented in the model are presented in detail.

Chapters IV, V, and VI address the major objectives of this research. A network performance analysis is discussed in Chapter IV. The design, analysis, and results of the simulator experimentation are presented. Chapter V expounds the design and development of a model that uses performance data to dynamically reconfigure the topology of an integrated network. The properties of the model with regard to logical and functional modularity are also described. Chapter VI illustrates the topological optimization of integrated networks by displaying the results of applying the model to several sets of design criteria.

Finally, a summary of the results of this research and proposed recommendations for further research are presented in Chapter VII.

The appendixes are attached to aid the potential user of the model. The model and its associated documentation are given, along with sample input and output.

CHAPTER II

LITERATURE SURVEY

2. Introduction

The theory and practice of computer networks has evolved in parallel, and networks were built before a clear understanding of such networks was achieved [60, 75]. And although the understanding of network design and performance has improved considerably over the years, many of the early problems are still receiving a great deal of attention. Among these are switching, routing, flow control, performance evaluation, and topology. Each of these areas is a complex study in itself, not to mention the fact that they are all interrelated. Therefore, this survey will examine and highlight each topic only as far as is necessary in order to support or explain portions of this research.

2.1 Switching Techniques

Computer-communication networks are classified in a variety of ways [3, 28, 74, 92, 95]. One such criterion is the communications technology that is used to transfer information from one link in a network to an adjacent link via a transmitting node. That is, the switching discipline used is generally a useful and accurate

descriptor of computer networks. Early networks relied on one of two switching approaches: circuit switching or store-and-forward switching. Each of these disciplines has advantages and disadvantages; and as the communications technology improved and new ideas evolved, several other switching techniques surfaced. The following paragraphs describe the four most commonly-referenced switching disciplines to date.

2.1.1 Circuit Switching

In a circuit-switched network, a physical path must be set up between the source node and the destination node prior to the start of information transfer. This path may traverse intermediate switching points (nodes). Users compete for the resources which, once acquired, remain dedicated for the duration of the transaction. Once the session is complete, the switching equipment disassembles the path and these resources are once again made available to the pool of users.

Because of its low line overhead and minimal delay [27, 43, 56], circuit switching is particularly useful in applications characterized by a steady, non-bursty flow of information. The telephone networks of the United States are circuit-switched systems. Users do not have dedicated voice channels but compete for limited resources. Circuit switching has remained the most effective approach for

voice users [96], but early attempts by data users to use circuit switching resulted in problems with switching delay times and circuit utilization.

2.1.2 Store-and-Forward (Message) Switching

A second approach, store-and-forward or message switching, is more data oriented. Using this approach, a total set of information (the message) is sent from one user in the system to another by establishing a link from the "sender" to a connected node. Once the node receives the message, it must store and log it for accounting purposes until it can be forwarded to the next node in the routing scheme. Once the message is stored, the node releases the communications link. This process is repeated from node to node until the message is accepted by the "receiving" node. An acknowledgement is usually sent back through the system. Some message-switched systems break up the message into fixed blocks of information. If this is the case, then all of the message blocks must be received in the prescribed order at an intermediate node before the message can be retransmitted.

While circuit switching results in a fixed message transmission delay, message switching involves variable transmission delay. Message switching with fixed routing has been used in many commercial applications (e.g.,

Control Data Corporation's CYBERNET). It is also the approach of DOD's AUTODIN data network. Limited adaptive routing schemes allowed store-and-forward systems to better utilize their links by allowing routing decisions to be made dynamically at intermediate nodes; however, in many cases, link utilization was less than effective and the nodal storage requirements and message manipulation times (logging, storing, and transmitting) were often excessive. As a result, an improved switching philosophy, packet switching, emerged.

2.1.3 Packet Switching

Using packet switching, the system breaks a message into fixed size "packets" (although some variable length packet schemes exist). Each packet is transmitted from source to receiver over an available link out of each intermediate node. The packets are not required to arrive at their destination in any particular order since the message reconstruction process at the receiving node is independent of the order in which the packets are received. ARPANET, for many years the mecca of networking decisions, has shown that packet switching can improve link utilization, response time, and network throughput [77, 78]. However, packet switching is not without its problems. Applications and research have demonstrated that the lack of packet flow control can in some instances

cause disastrous problems. As a result, many constraints have been placed on packet-switched networks. For example, trace capabilities have been added to packets and senders may be required to reserve final destination storage prior to shipping the message. Nevertheless, packet switching has become a dominant force in data communication systems [78]. In fact Roberts, a former ARPANET project director, states that "experience with ARPANET has demonstrated that computing service can be obtained remotely through a computer network at one third the cost of a local dedicated system" [77].

2.1.4 Integrated Switching

The most recent approach to switching is the integrated or hybrid approach. In this technique both circuit-switched and packet-switched components are used in the network. There are a number of potential scenarios for integrating voice and data in the same network [5, 16, 18, 25, 33, 53, 55, 69, 71, 83, 84, 94, 96, 97]. According to Dysart and others [5, 16, 18, 25], one of the most promising techniques is to use circuit switching for voice traffic and packet switching for data traffic. The traffic is then integrated on a circuit-switched backbone.

2.2 Routing and Flow Control

Although a wide spectrum of routing schemes is

available, most can be categorized into deterministic or stochastic subgroups [36, 48, 74]. Deterministic schemes include subclassifications such as flooding, fixed, and ideal observer approaches [74]. Deterministic approaches usually have their decisions made at the nodes or end points. Stochastic schemes, on the other hand, are usually considered to be adaptive in nature. That is, they dynamically revise the routes based on current loads. The stochastic subgroup includes the distributed, isolated, and random techniques [74]. The approach selected will no doubt impact node complexity, the need for network control nodes, computational complexity, reliability, and reconfiguration capabilities [42, 43, 64]. Some approaches (e.g., the ideal observer) find uses in studying bounds or thresholds but find no practical implementations.

Flow control and congestion control are closely related and are often intertwined in the literature as well as in the implementation of approaches. For the purposes of this research, flow control is limited to those control procedures which attempt to regulate the system inputs during periods of congestion [10, 43]. Most control flow mechanisms place restrictions on the sender or allow the receiver to ignore messages at will. Whereas flow control is an end-to-end phenomenon, congestion control is concerned with insufficient buffer space in a

localized area. Flow control measures are typically used to reduce or eliminate congestion because they are easy to implement [41, 95]. Unfortunately, these actions oftentimes meet with failure due to the bursty nature of data communication.

2.3 Network Performance Evaluation and the Topological Approach

Performance evaluation is of concern to all engineering or design personnel. In networks, the complexity of the system design and the interrelationships of system components in a myriad of combinations makes any evaluation study a difficult task. Because of the complexity, the range of alternatives for consideration must be reduced to a workable set. Many different approaches and criteria have been studied. Wilkov [99] concentrates on "reliability and availability of communications paths between all pairs of centers in the network". Yet, he concludes that "almost all of the reliability measures treated to date do not reflect the degradation in network performance that results from failure of network elements in the presence of a specified traffic demand". Most studies, in fact, look to simulation or modeling [57]. One recent example [89] states that most work has been done using primarily "discrete event simulation models". It should be noted that most of this work has been in the area of

store-and-forward or packet type networks. The Value-Added Network Simulator (VANS) system [89], which models distributed, resource-sharing computer networks, is capable of simulating circuit or packet-switched networks, but it is not capable of simulating an integrated circuit/packet-switched network.

Chou and his coauthors [11, 14, 32] use the network topology to investigate the performance of a distributed computer network. They point out that the basic topological approach is concerned with specifying the "locations and capacity of communications links within the network... which satisfies constraints on response time, throughput, reliability, and other parameters". Chou et al. [11, 14, 32] suggest a number of methods that can be used to analyze the performance of a distributed network via the system topology. One such method is an iterative, heuristic technique. This technique consists of "specifying a feasible 'starting' network" from which the model iteratively uses a heuristic scheme to modify the network in simplified steps until the performance constraints are satisfied. Once the process has satisfied the constraints, it tries to continue improving until it is no longer possible to do so. It is then said to be "locally optimal" [32]. Chou also points out that the question of routing and flow control adds to the complexity and time required for computation in large

network models.

Chou has recently developed the ACK/TOPS model [11] which implements an iterative, heuristic scheme to "determine the least cost network design satisfying traffic and response time requirements". Although this model, which was privately developed, addresses network topologies that are tree-shaped, ring-shaped, or have packet-switched backbones (i.e., subnets), the model does not apply to integrated circuit/packet-switched networks. In fact, Chou says [13] that substantial modifications would have to be made to his model in order to incorporate the capability of handling any kind of voice or integrated network.

As used here, the term "integrated circuit/ packet-switched network" denotes a distributed computer network possessing a circuit-switched backbone or subnet with numerous packet-switched local access networks feeding into the communications subnet. Hsieh et al. [53] and Rudin [84] both emphasize that the integrated circuit/packet-switched network as defined above will become more prevalent in the future, eventually replacing circuit-switched or packet-switched systems. They cite the heavy research interest in "hybrid switching", as well as DOD's proposed Defense Communication System [53] as the trend setters for the future.

CHAPTER III

THE NETWORK SIMULATION MODEL

3. Introduction

This chapter describes the simulator that is used to generate performance data for integrated networks. The simulator is a modified version of the network simulation model developed by Clabaugh [15]. Modifications were made to increase the flexibility of the model so that it could be used in a dynamic network topology reconfiguration scheme. The major changes incorporated in the model fall into two categories:

(1) a variable link capacity, and

(2) an expanded statistics gathering capability.

The documentation for the simulation model given in Appendix A of Clabaugh [15] has been updated to reflect all of the incorporated changes and is attached as Appendix C in this dissertation. The description that follows is based on Clabaugh's work [15].

3.1 Description of the Integration Concept

The model developed in the simulator depicts an integrated circuit/packet-switched network that consists of the following major components (Fig. 1):

A. Backbone Circuit Switch (CS) Nodes

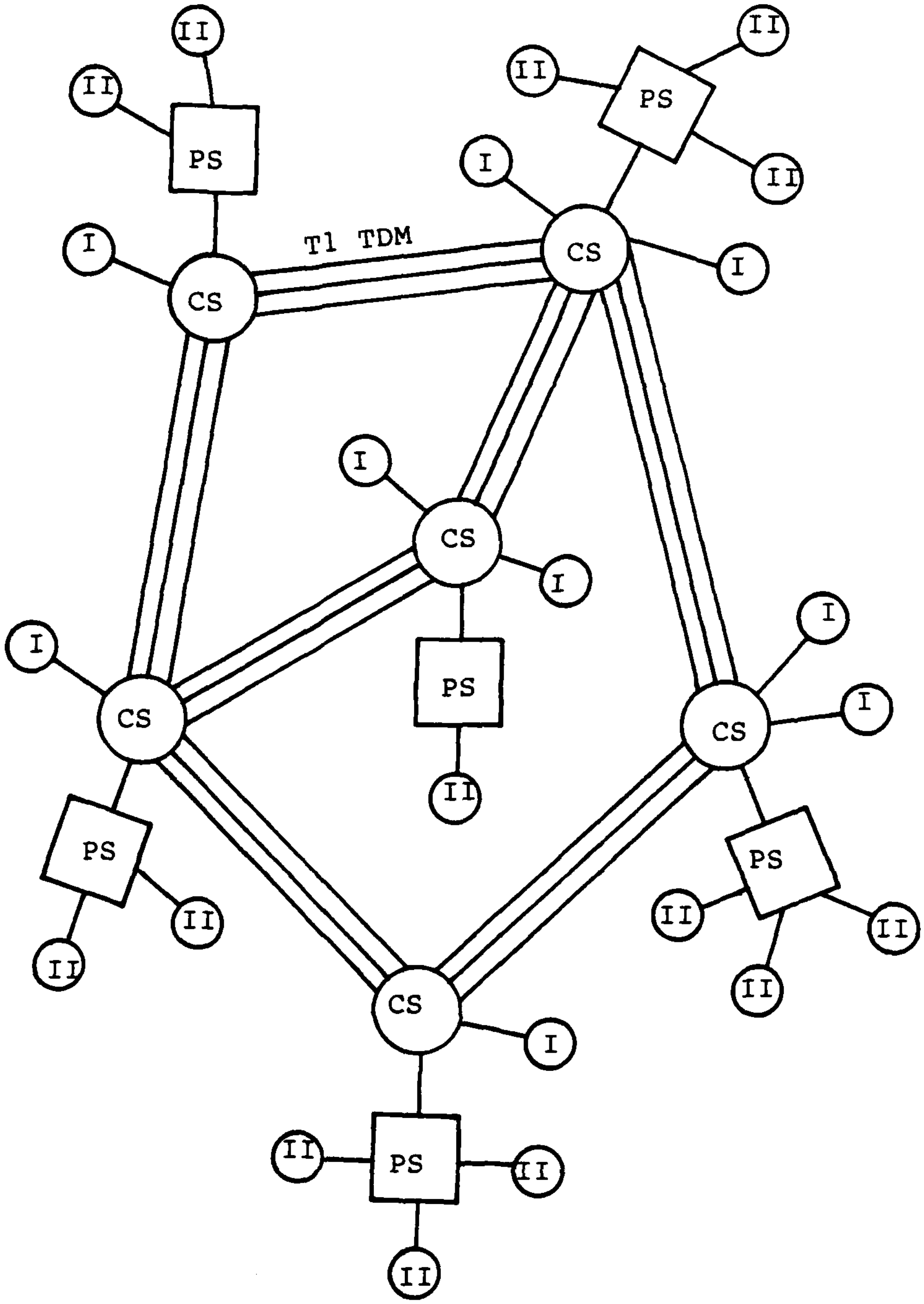


Figure 1. Integrated network configuration

B. Peripheral Packet Switch (PS) Nodes

C. Invariant Network Synchronous Time-Division-Multiplexed (TDM) Frame Switching Superstructure

D. Digital Network Using T1 Carriers and Digital Switching Nodes

E. Variable Subscriber Data Rates

F. Two Classes of Subscriber Traffic

1. Class I: Real-time traffic that once started cannot be interrupted (voice, video, facsimile, and sensor).

2. Class II: The general class of packet data, such as interactive, bulk, and narrative/message.

The backbone CS nodes and peripheral PS nodes form the nucleus of a distributed computer-communication network in which the transmission of data and voice between any two nodes on the subnet is accomplished by sharing the capacity of the T1 link. A Slotted Envelope Network (SENET) self synchronizing concept [18] is used to achieve simultaneous transfer of voice and data on the carrier. This concept treats the available bandwidth on a digital link as a resource for which all forms of communication must compete. Using SENET, the T1 link is synchronously clocked into frames of a fixed time duration, b, which are assumed invariant throughout the network. Each frame is partitioned into several data

slots (channels) for which the various traffic types compete (Fig. 2). The self synchronizing capability within each frame is implemented by using a few bits as a start-of-frame (SOF) marker to indicate the beginning of each of a contiguous series of constant period frames.

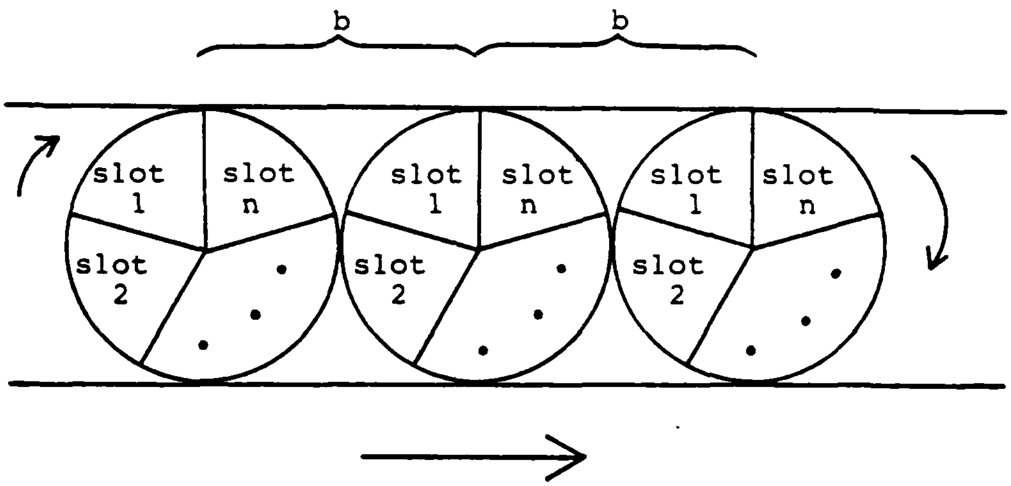


Figure 2. SENET frame clocking

3.2 Description of the Routing Doctrine

Voice (Class I) subscribers are terminated directly at circuit switches to avoid packetizing and any unnecessary routing overhead through packet switches. Similar to a telephone dial-up process, a Class I transaction results in a physical end-to-end connection for the duration of the call or a system loss (blocking) occurs. Although not a design requirement, packet switches are collocated with the circuit switches. All

data (Class II) subscribers are terminated at packet switches. While the Class II subscriber packet switch interface depends upon the individual terminal hardware configuration, the transmission of data between the packet and circuit switches is accomplished using TDM. The packet switches are primarily responsible for managing the movement of packets between input terminals and the circuit switches; placing traffic in queues according to a regional routing policy; and performing connection initiation, circuit disconnect, and coordination with other packet switches, depending on network loading.

The regional routing doctrine for each packet switch, coupled with virtual switch connections, reduces overhead and the traffic congestion problem. As traffic is entered into a packet switch from subscriber terminals, it is queued for the relevant destination packet switch. A circuit switch connection is then initiated/terminated by the packet switch on behalf of this traffic. A circuit switch connection can be established on a single transaction basis, similar to an interactive communication, or on a multiple transaction basis if the traffic is bulk data, message/narrative traffic, or if several users are queued for the same destination packet switch. This routing scheme (1) insures minimal queue build-up within the backbone, and (2) enforces an end-to-end flow control strategy.

Progressive alternate routing is used on the backbone of the network. With this method, each circuit switch node has a primary and an alternate path. If blocking occurs at some node during connection initiation, the alternate route is tried for route completion. If this connection fails, the transaction is either queued at the packet node or considered a system loss at the circuit node, depending on its class.

3.3 Description of the Queueing Concept

For the integrated computer-communication network described, the numerous internodal conditions and variables preclude any exact analytic solution. However, by decomposing the network into nodal queueing processes, the simulation model can be viewed as a system of simple queueing models.

The traffic flow at each packet switch can be described as follows:

1. Each Class II subscriber communicates with the packet switch via independent, Poisson transaction arrivals and exponentially distributed transaction interarrival times.
2. The message lengths (packets per message) are assumed to be geometrically distributed. This conforms to the study of multiaccess computer communications by Fuchs and Jackson [35].

3. Each packet switch can be thought of as a $M/M/C$[1] system (Kendall notation) [49], with infinite storage.
4. Packets are placed on the packet switch queue and served on a first-come-first-served (FCFS) basis.

The traffic flow entering each circuit switch node originates at either neighboring circuit switch nodes, connected packet switch nodes, or locally terminated Class I subscribers. Since all traffic entering from other than terminated Class I subscribers see a physical connection, only the Class I subscribers enter into a serving mechanism process at the circuit switch node. These subscribers are assumed to possess Poisson arrival and exponential service distributions. Thus, the M/M/C/C queueing model suffices to represent this network model. Both the PS and CS queueing systems are impacted by channel availability, since the model policy forces data and voice subscribers to compete for the available slots.

1 In this notation the first element denotes the interarrival time distribution, the second element denotes the service time or message length distribution, and the third element denotes the number of servers. If a fourth element is given, it denotes the queue or buffer size available. M stands for the (Markov) exponential distribution, G for a general or arbitrary distribution, and D for the (deterministic) assumption of constant interarrival or service times.

In summary, delays and queues at each node are approximated closely by $M/M/C/\infty$ and $M/M/C/C$ queueing processes (Fig. 3). Since each of these processes is globally impacted by channel availability, the network simulation provides performance measures for end-to-end packet delay and voice call blocking.

3.4 Properties of the Model

Since the communication activity is centered around the nodes in a network, the model is said to be node-based. There are three principal nodal tables: routing, channel, and queue tables. The routing tables are used to determine the output channel (link) a transaction will take from a given node in its attempt to reach its destination. Since each T1 link is full-duplex (FDX), i.e., traffic can move in both directions simultaneously, the link is represented as two independent half-duplex (HDX) channels. Information stored and updated within the various channel table entries for each node is used in the gathering of such performance statistics as link utilization and throughput. The nodal queue tables are used to obtain statistics associated with average transaction time in the system and various transaction oriented delay and blocking statistics. A complete description of each of the tables used in the model is given in Appendix C. From these descriptions it

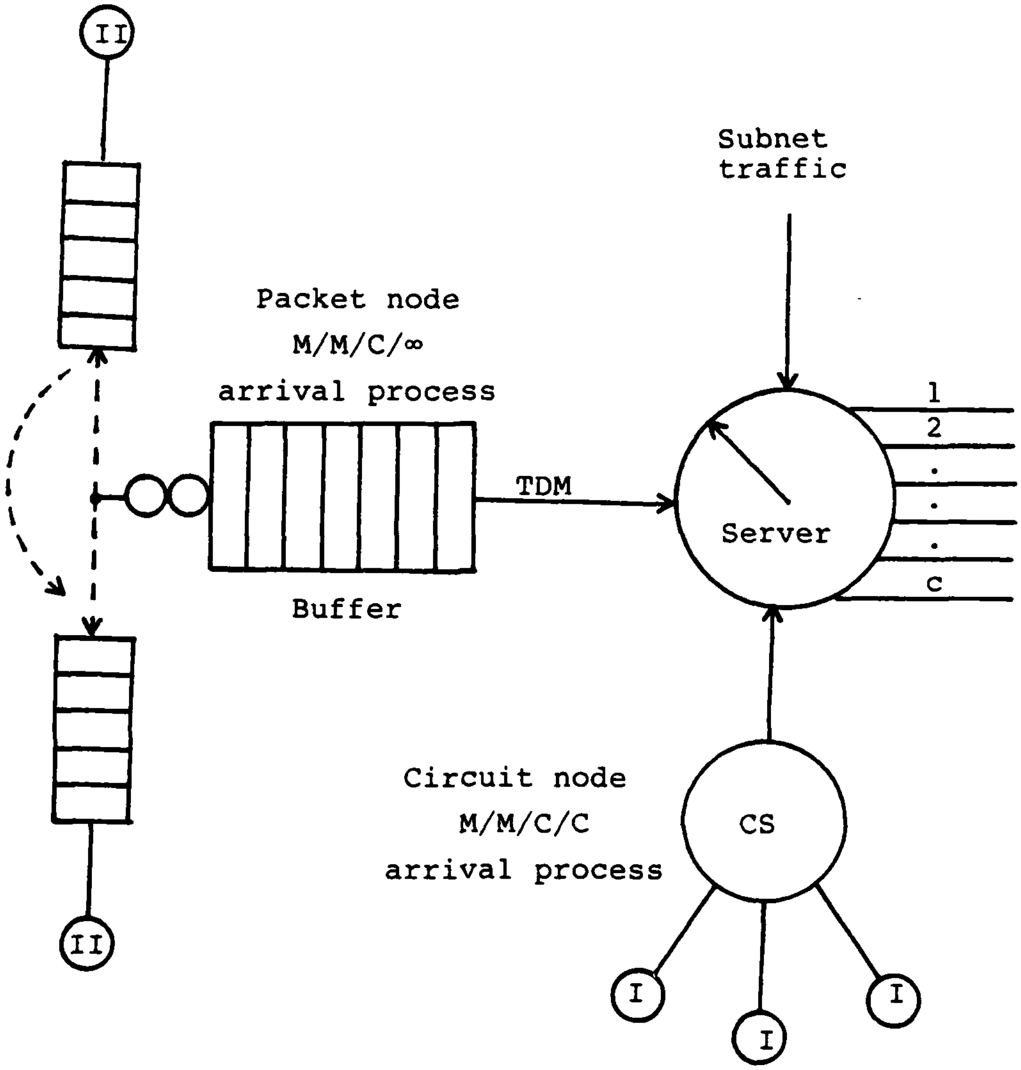


Figure 3. Network nodal queueing processes

is apparent that the model is capable of generating a wide variety of performance statistics.

An event table reflects network status disturbing events. Data and voice transaction arrivals or departures at a node are examples of event table entries. The simulation is driven by event changes that occur at each node.

3.4.1 Simulator Input Parameters

The network simulation model requires that the following parameters be input to the model.

(1) Number of Nodes

(2) Number of Links

(3) Number of Time Slots for Each Link

(4) Number of Slots Required by a Data Packet

(5) Frame Time

(6) Nodal Switching Delay

(7) CS Arrival Rate

(8) PS Arrival Rate

(9) Simulation Start Time

(10) Simulation End Time

(11) PS Saturation Level Indicator

(12) Voice Digitization Rate

(13) PS Buffer Size

(14) Voice Call Service Rate

(15) Number of Bits per Packet

(16) Average Number of Packets per Message

Due to the large number of system input parameters, the simulator can generate empirical data for an enormous number of topology/workload combinations. The network traffic load is determined by the voice and data arrival rates (parameters 7 and 8) and the voice call service rate (parameter 14). Link capacity is seen to be a function of parameters 3, 4, 5, and 15. The exact relationship is given by

Link Capacity =

$$\frac{1000 * PARAM(3) * PARAM(15)}{PARAM(4) * PARAM(5)} \text{ bits/second(bps).}$$

The input parameters are described in more detail in Appendix C.

3.4.2 Network Performance Measures

For each possible combination of input parameters, the simulator is capable of generating more statistics than one might care to examine. The user has two statistical routines available and can have cumulative statistics printed at regular time intervals if desired. Examination of the code and table descriptions in Appendixes A and C will reveal a myriad of possible output choices. The following five performance measures that the model generates are listed here because these represent the performance measures that are most commonly seen in

the literature:

(1) Mean Packet Delay (MPD)

(2) Average Link Utilization (ALU)

(3) Packet Throughput (THR)

(4) Average Queue Length (AQL)

(5) Fraction of Calls Blocked (BLK)

Specification of bounds on any combination of these measures establishes what is generally called a grade-of-service (GOS) for the network.

CHAPTER IV

NETWORK PERFORMANCE ANALYSIS

4. Introduction

The verification and validation of simulation models is a difficult task [47, 91]. In the absence of real data from the system being simulated, the task seems even close to insurmountable. Action has been taken, however, to gain a high confidence level in the simulator described in Chapter III.

Verification of simulator performance with regard to voice/data transaction arrival patterns was accomplished by Clabaugh [15]. He found that the simulator behavior was within a 95% confidence interval of the true means, where the true means were determined by the use of standard Poisson generators and user run time specifications. Clabaugh also provides additional verification of simulator output by configuring the simulator to correspond to an analytic model and comparing the output results [15]. These verification efforts apply to the modified model as well, since none of the modifications impact Clabaugh's verification work. This chapter documents another step taken toward the verification and validation of the integrated network simulator.

The motivation for this analysis stems not only from the desire to move in the direction of model verification but also from the need to obtain and analyze performance data from an integrated circuit/packet-switched computer network, for such data is scarce. Fortunately, these two encompassing goals do not diverge.

4.1 Goals and Scope of the Analysis

The specific goals of the analysis are as follows:

(1) Design an experiment whereby the performance data can be efficiently and economically obtained.

(2) Determine the effective ranges of network parameters for which realistic and acceptable network performance results.

(3) Investigate the sensitivity of performance measures to changes in the network input parameters. Specifically, determine how network performance is affected by changes in the network traffic load, trunk line or link capacity, and network size.

The scope of the analysis is restricted to those parameters that are closely related to the network topology and the workload imposed on that topology. With regard to performance sensitivity to workload and link capacity, the following four parameters are investigated:

CS: Circuit Switch Arrival Rate (voice calls/min)

PS: Packet Switch Arrival Rate (packets/sec)

SERV: Voice Call Service Rate (sec)

SLOTS: Number of Time Slots per Link (a capacity indicator)

The sensitivity to network size is also investigated by varying the number of nodes and links in a network. In all cases, the performance measures observed are mean packet delay (MPD), fraction of voice calls blocked (BLK), and average link utilization (ALU).

4.2 Design of Experiment

The 10-node network topology shown in Figure 4 was considered sufficiently complex to provide practical performance data without exhausting the computing budget.

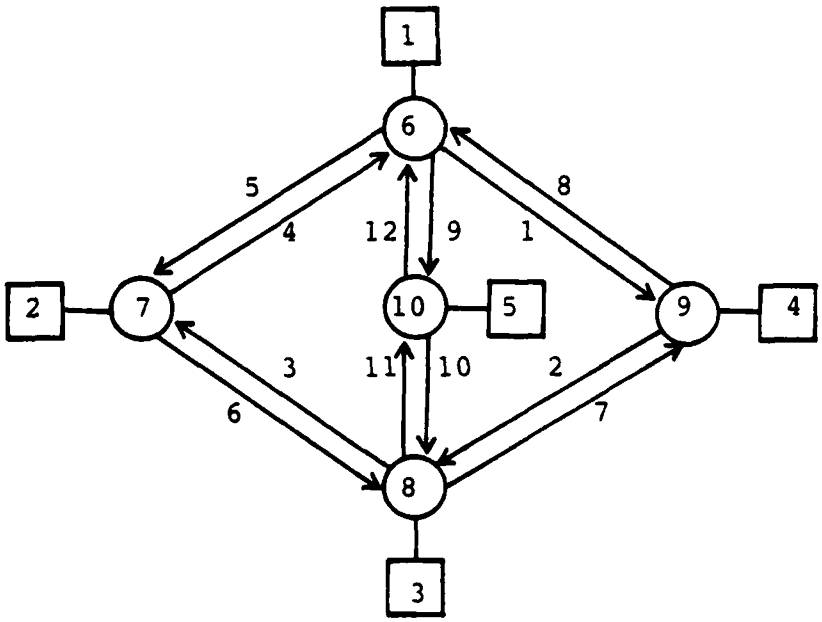


Figure 4. 10-Node network topology

The circuit switch (circular) nodes correspond to the major computing centers of Tymshare's TYMNET, a circuit-switched network [74]. The trunk lines are full-duplex (FDX) carriers, each of which is modeled by two independent, half-duplex (HDX) channels.

A preliminary sizing analysis [73] indicated that the effective range of input parameters to be investigated is as follows:

CS: 0-8 (calls/minute at each node)

PS: 0-600 (packets/sec at each node)

SERV: 60-300 (sec/call)

SLOTS: 28-52 (a link capacity indicator)

The fixed parameter settings were such that each slot represents a capacity of about 33 Kbits/sec.

The experimental design selected for this analysis is a second order (quadratic), rotatable, central composite design [67, 68]. Such a design for k (number of parameters) = 3 is illustrated in Figure 5. This design was chosen because it reduces considerably the number of experimentation points that would otherwise be required if the classical 3^k factorial design were used. The "central composite" feature of the design replaces a 3^k factorial design with a 2^k factorial system augmented by a set of axial points together with one or more center points. A "rotatable" design is one in which the prediction variance is a function only of the distance

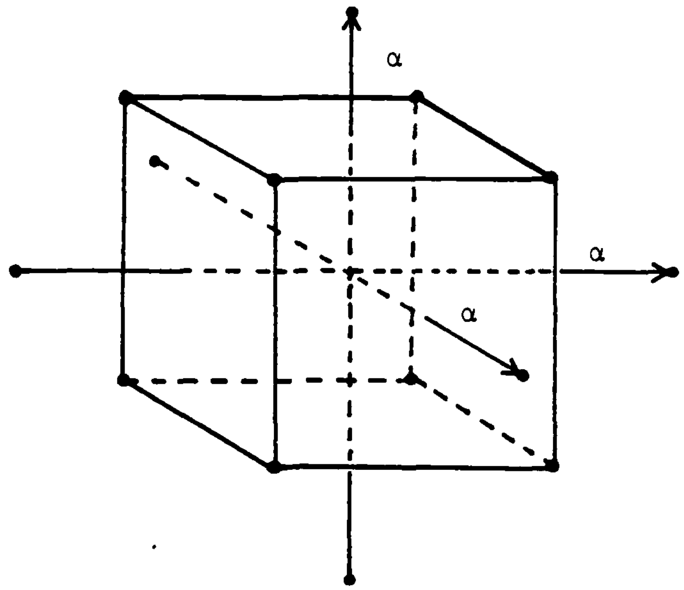


Figure 5. Central composite design (k=3)

from the center of the design and not on the direction. Myers [67] shows that the number of experimental points required for k = 4 is 31. That is, there are 16 factorial points, 8 axial points, and 7 replications at the center point. In order for the design to be rotatable, Myers [67] shows that α must be chosen as $(2^k)^{1/4}$. In this case, α, the distance from the center point to an axial point, is 2. The seven replications at the center point will allow an estimate of the experimental (pure) error to be made; thus, a check for model adequacy is possible [70]. Myers [67] recommends seven replications

at the center point simply because it results in a near-uniform precision design. That is, it is a design for which the precision on the predicted value $\hat{y}$, given by

$$\frac{N \ \mathrm{var}\ (\hat{y})}{\sigma^2} \quad , \text{ where}$$

N = total number of experimental points, and

σ^2 = the error variance,

is nearly the same at a distance of 1 from the center point as it is at the center point [67].

In this design, each parameter is evaluated at five different levels. The five levels for each of the four parameters are as follows:

CS:	0	2	4	6	8
PS:	0	150	300	450	600
SERV:	60	120	180	240	300
SLOTS:	28	34	40	46	52

The center point is defined as CS/PS/SERV/SLOTS = 4/300/180/40.

Analysis of the results of these 31 runs indicated that the original range of data was too large. Ten of these experimental data points resulted in network performance that would be totally unacceptable, e.g., a MPD of more than 10 seconds. These "outliers" were eliminated and 14 additional points in the moderate to heavy loading of the network were added. Two observations

were taken at 8 of these 14 additional points for the purpose of estimating the error variance at points other than the center of the design. Table I shows the 43 observations that form the data set used in the subsequent regression analysis. The first 21 observations in Table I represent those data points retained from the original 31-run design. The remaining 22 observations represent the data points used to augment the original design.

Table I. Experimental data

OBS	CS	PS	SERV	SLOTS	MPD	ALU	BLK
1	2	150	120	34	0.117	0.262	0.000
2	2	150	120	46	0.116	0.194	0.000
3	2	150	240	34	0.113	0.372	0.000
4	2	150	240	46	0.116	0.275	0.000
5	2	450	120	34	0.430	0.553	0.016
6	2	450	120	46	0.260	0.396	0.000
7	2	450	240	34	0.433	0.676	0.031
8	2	450	240	46	0.449	0.481	0.003
9	6	150	120	34	0.339	0.520	0.012
10	6	150	120	46	0.115	0.386	0.000
11	6	450	120	46	0.462	0.594	0.016
12	0	300	180	40	0.123	0.231	0.000
13	4	0	180	40	0.000	0.325	0.000
14	4	300	180	52	0.153	0.427	0.000
15	4	300	180	40	0.350	0.561	0.016
16	4	300	180	40	0.527	0.571	0.005
17	4	300	180	40	0.502	0.561	0.011
18	4	300	180	40	0.316	0.581	0.020
19	4	300	180	40	0.381	0.562	0.017
20	4	300	180	40	0.798	0.608	0.036
21	4	300	180	40	0.616	0.560	0.007
22	5	400	210	43	1.613	0.739	0.070
23	5	400	210	43	2.754	0.746	0.082
24	5	400	150	37	1.656	0.725	0.067
25	5	400	150	37	1.345	0.725	0.084
26	5	400	150	43	0.192	0.615	0.027
27	5	400	210	40	4.814	0.785	0.106
28	5	400	210	40	5.624	0.782	0.125
29	4	400	210	37	1.604	0.747	0.084
30	4	400	210	37	2.730	0.745	0.101
31	4	400	210	40	2.440	0.695	0.063
32	4	400	210	40	1.199	0.687	0.057
33	4	400	180	40	0.910	0.648	0.035
34	5	400	180	40	1.970	0.722	0.074
35	5	400	180	40	1.130	0.731	0.077
36	5	400	180	37	4.084	0.775	0.110
37	5	400	180	37	4.019	0.778	0.139
38	4	400	180	37	4.817	0.712	0.057
39	4	400	180	37	1.135	0.701	0.065
40	3	400	210	37	0.599	0.652	0.017
41	3	400	210	43	0.293	0.541	0.003
42	3	400	150	37	0.454	0.570	0.003
43	3	400	150	43	0.280	0.476	0.005

4.3 Regression Analysis and Model Selection

The Statistical Analysis System (SAS) [85, 86, 87] has been used to perform a multiple regression analysis for each of the three performance measures. The regression variables are the four input parameters. SAS procedures REG [86] and RSREG [86] were used to analyze the data. The assumption of a quadratic response surface allows for the estimation of 15 model parameters, including the intercept.

Although the model selection procedure can be based on a number of possible criteria [23, 34, 66], the approach taken is as follows. RSREG was used first to check the full (quadratic) model for specification error (lack of fit test) and to determine significance levels for the linear, quadratic, and crossproduct terms. The models for MPD, BLK, and ALU exhibited a lack of fit that was significant at the .14, .05, and .82 levels, respectively. It was found, however, that these significance levels were heavily influenced by the observations at the extremes of the heavy-loading region. For example, by eliminating 13 of the observations in the heavy-loading range of data, the lack of fit for BLK could be raised to the .70 significance level. Hence, although the .05 level for the BLK model borders on statistical significance, the lack of fit was not deemed sufficient to justify a more complex model.

The RSREG results also indicated that some terms were insignificant and possibly could be deleted from the model. Using the RSREG results as a guide, several subsets of the full quadratic models were investigated using SAS procedure REG. In these models, all linear terms were retained because even though a lower order term in a polynomial model may not be considered significant, dropping such a term could produce a misleading model [34]. Several crossproduct and quadratic terms were deleted, however, and the final regression models selected are shown in Table II. Despite their appearance of being insignificant, several of the crossproduct and quadratic terms were retained in the model simply because their retention resulted in a smaller residual mean square than if they had been deleted. The interpretation of these regression models is presented graphically in the following section in terms of a sensitivity analysis.

4.4 Sensitivity Analysis

This section presents data to describe how the various performance measures change with respect to changes in traffic load, link capacity, and network size. All of the graphs presented are obtained from the quadratic response surfaces (regression models) developed in the previous section.

Table II. Regression models

DEP VARIABLE: MPD

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	10	61.561986	6.156199	6.874	0.0001
ERROR	32	28.658932	0.895592		
C TOTAL	42	90.220918			
ROOT MSE		0.946357	R-SQUARE	0.6823	
DEP MEAN		1.218093	ADJ R-SQ	0.5831	
C.V.		77.69169			

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR HO: PARAMETER=0	PROB > |T|
INTERCEP	1	12.556905	13.142988	0.955	0.3465
X1 CS	1	-1.654903	1.326989	-1.247	0.2214
X2 PS	1	0.014620	0.012349	1.184	0.2452
X3 SERV	1	-0.026635	0.010541	-2.527	0.0167
X4 SLOTS	1	-0.531166	0.597025	-0.890	0.3803
X11	1	0.232675	0.078122	2.978	0.0055
X12	1	0.001153381	0.0009976058	1.156	0.2562
X13	1	0.013125	0.00331975	3.954	0.0004
X14	1	-0.048214	0.025537	-1.888	0.0681
X24	1	-0.000395179	0.0003018693	-1.309	0.1998
X44	1	0.009018565	0.007302992	1.235	0.2259

DEP VARIABLE: ALU

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	11	1.123895	0.102172	970.909	0.0001
ERROR	31	0.003262242	0.0001052336		
C TOTAL	42	1.127158			
ROOT MSE		0.010258	R-SQUARE	0.9971	
DEP MEAN		0.581233	ADJ R-SQ	0.9961	
C.V.		1.764929			

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR HO: PARAMETER=0	PROB > |T|
INTERCEP	1	0.022183	0.150510	0.147	0.8838
X1 CS	1	0.068950	0.014713	4.686	0.0001
X2 PS	1	0.001859007	0.0001334741	13.928	0.0001
X3 SERV	1	0.001788392	0.0006903726	2.590	0.0145
X4 SLOTS	1	-0.0094644	0.006917257	-1.368	0.1811
X11	1	-0.000929364	0.0008468003	-1.098	0.2809
X13	1	0.0003897799	.00003664077	10.638	0.0001
X14	1	-0.00127589	0.0002929912	-4.355	0.0001
X24	1	-.0000259887	.00000329748	-7.881	0.0001
X33	1	-.0000024518	.00000146947	-1.669	0.1053
X34	1	-.0000214946	.0000093248 1	-2.305	0.0280
X44	1	0.0001583866	.00008483328	1.867	0.0714

Table II. (Continued)

DEP VARIABLE: BLK

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	12	0.062669	0.005222451	26.743	0.0001
ERROR	30	0.005858444	0.0001952815		
C TOTAL	42	0.068528			
ROOT MSE		0.013974	R-SQUARE	0.9145	
DEP MEAN		0.038163	ADJ R-SQ	0.8803	
C.V.		36.61764			

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR HO: PARAMETER=0	PROB > |T|
INTERCEP	1	0.197892	0.205845	0.961	0.3441
X1 CS	1	-0.028562	0.020039	-1.425	0.1644
X2 PS	1	0.0003439953	0.0002038975	1.687	0.1020
X3 SERV	1	0.0001307622	0.0005157113	0.254	0.8016
X4 SLOTS	1	-0.012008	0.00887909	-1.352	0.1863
X11	1	0.006553606	0.00115578	5.670	0.0001
X12	1	.00005438269	.00001502268	3.620	0.0011
X13	1	0.0003339836	0.0000496682	6.724	0.0001
X14	1	-0.00175778	0.0004020761	-4.372	0.0001
X22	1	3.79530E-07	1.75917E-07	2.157	0.0391
X24	1	-.0000158292	.00000450668	-3.512	0.0014
X34	1	-.0000194136	.00001274852	-1.523	0.1383
X44	1	0.0002843414	0.0001092323	2.603	0.0142

4.4.1 Sensitivity to Traffic Load

The workload imposed upon an integrated network is described by the voice arrival rates (CS) at each circuit switch node and the data packet arrival rates (PS) at each packet switch node, as well as the length of service for each voice call (SERV). It is assumed that, for a given arrival at any particular node, all of the other nodes of the same type are equally likely to be the destination node for that arrival. That is, the workload is said to be uniformly distributed between node pairs. Unless otherwise stated, it is also assumed that SERV = 180

seconds. Besides these three parameters, the link capacity (SLOTS) parameter also affects the network traffic load. If the CS and PS arrival rates are also assumed to be fixed, then the smaller values of SLOTS represent a heavier network load while the larger SLOTS values correspond to a lighter network load.

Figures 6, 7, and 8 depict the MPD, BLK, and ALU performance measures, respectively, as a function of CS for four different SLOTS/PS combinations. Similarly, Figures 9, 10, and 11 show MPD, BLK, and ALU, respectively, as a function of PS for four load levels defined by combinations of SLOTS and CS. The performance measure sensitivity to the traffic load parameters CS, PS, and SLOTS (for SERV=180) is seen by observing the relative slopes and ordinate values of the four curves in each of the Figures 6 - 11. For example, examination of the curves in Figures 6 and 9 indicates that, within the range of data shown, MPD is more sensitive to CS than to either SLOTS or PS. Additionally, the sensitivity of MPD to both CS and SERV is shown in the three dimensional plot of Figure 12, where the higher levels of one input parameter are seen to magnify the effects of the other parameter.

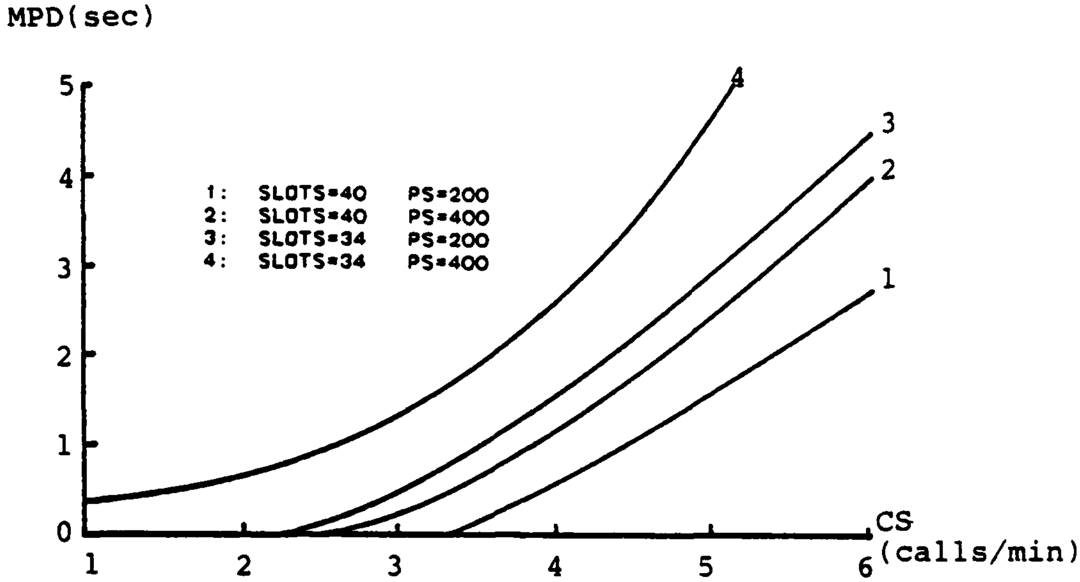


Figure 6. Mean packet delay (MPD) vs. voice arrival rate (CS)

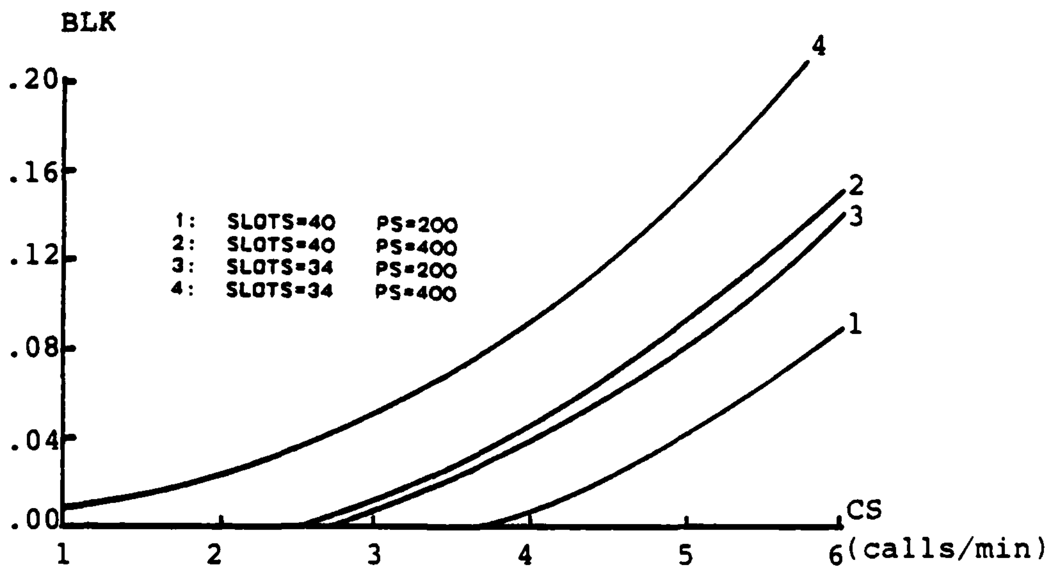


Figure 7. Fraction of calls blocked (BLK) vs. voice arrival rate (CS)

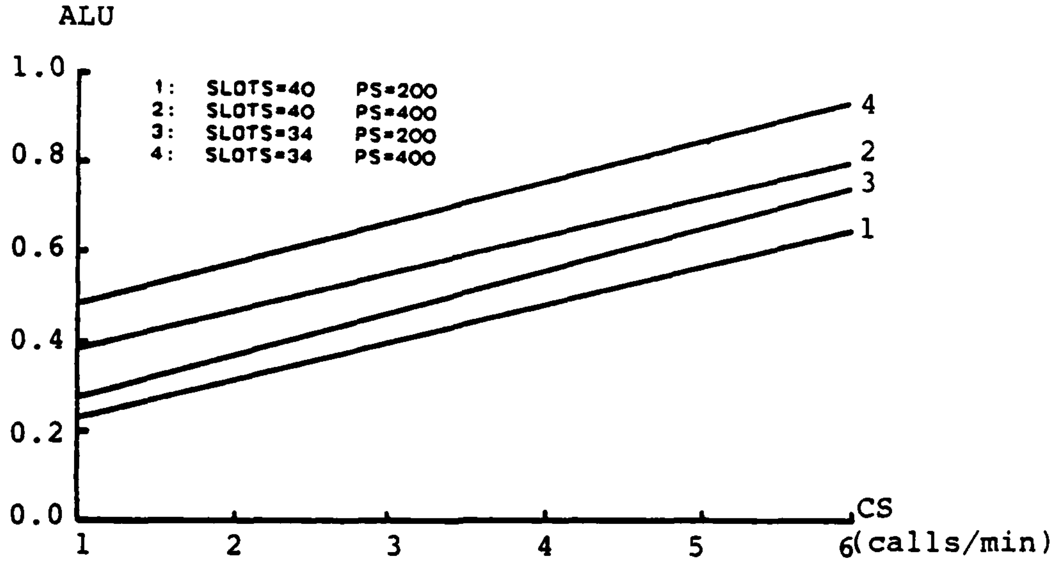


Figure 8. Average link utilization (ALU) vs. voice arrival rate (CS)

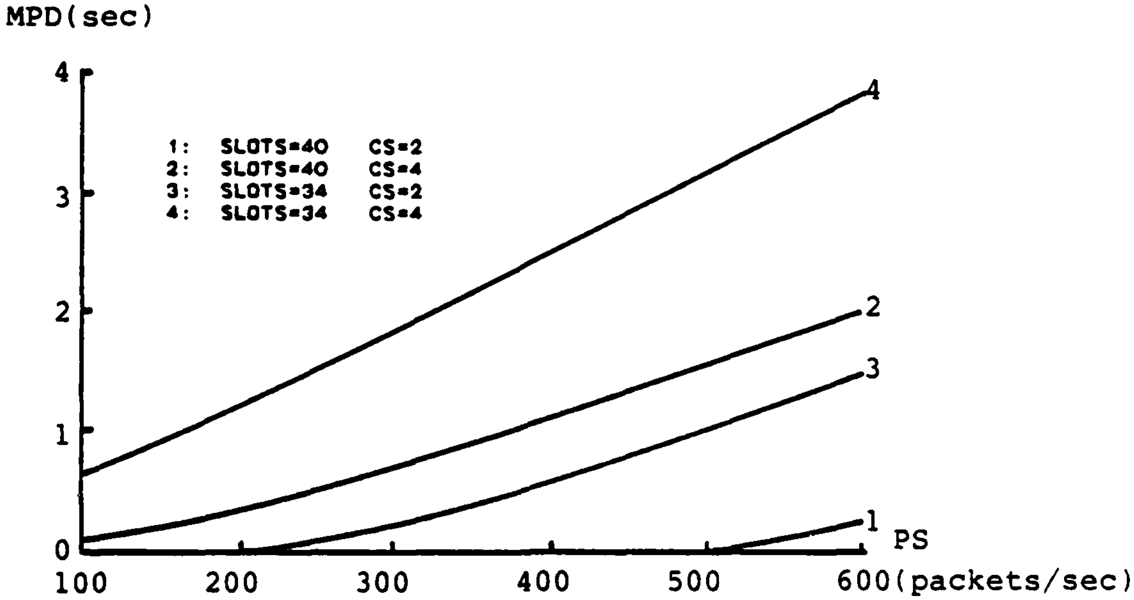


Figure 9. Mean packet delay (MPD) vs. data arrival rate (PS)

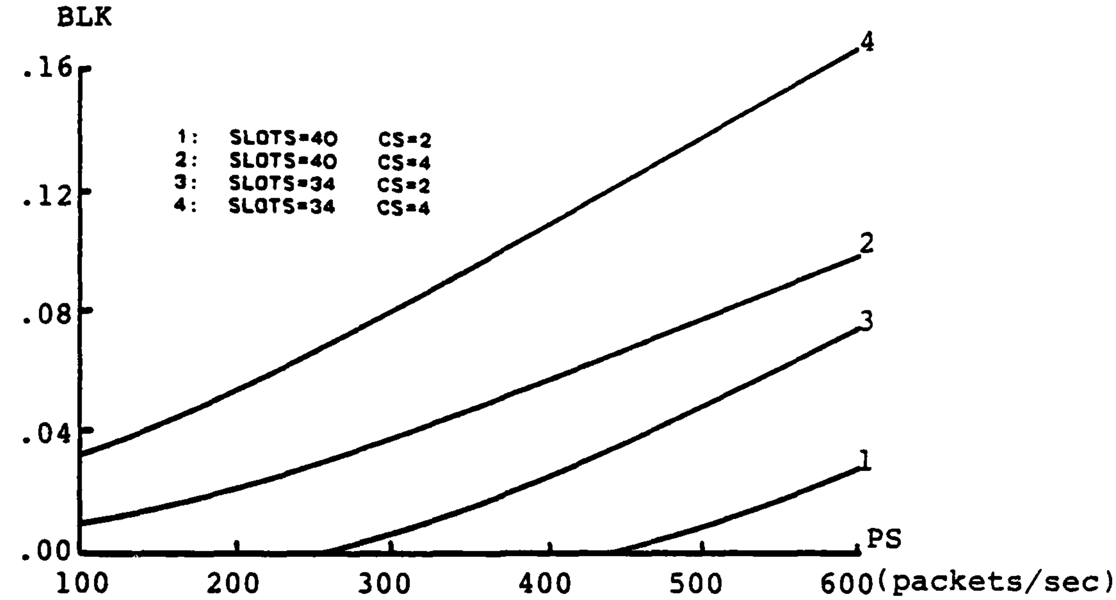


Figure 10. Fraction of calls blocked (BLK) vs. data arrival rate (PS)

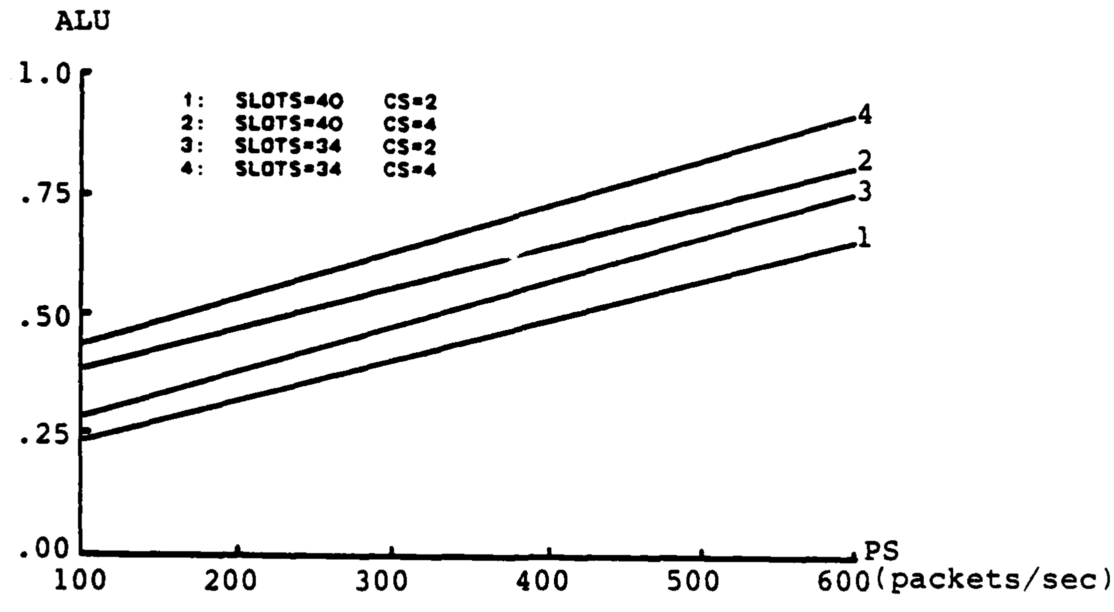


Figure 11. Average link utilization (ALU) vs. data arrival rate (PS)

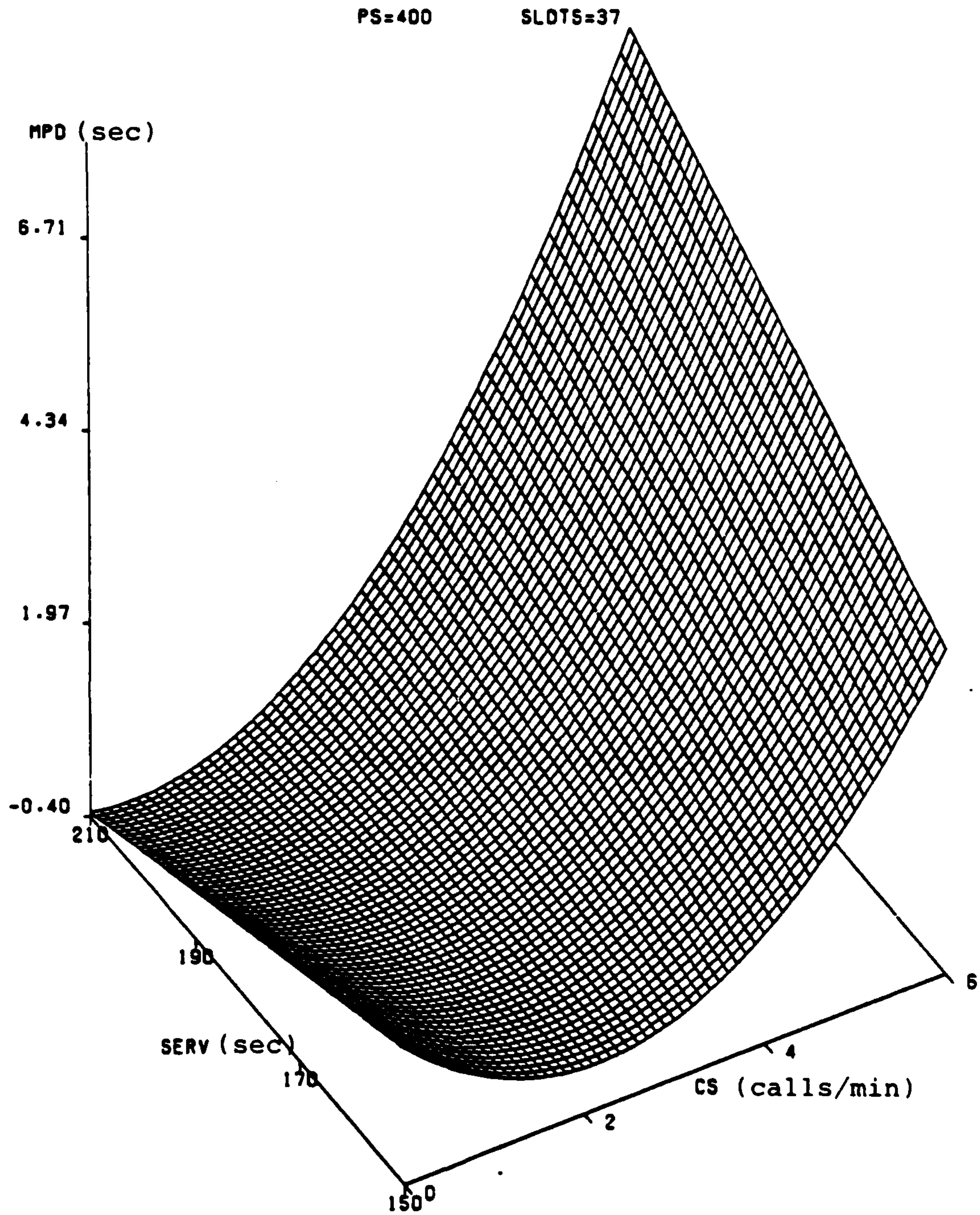


Figure 12. Mean packet delay (MPD) vs. voice arrival rate (CS) vs. voice service time (SERV)

4.4.2 Sensitivity to Link Capacity

The trunk line carrying capacity is an important design parameter in integrated networks. Hence, in addition to the plots of the performance measures versus number of SLOTS, which are shown in Figures 13, 14, and 15, the confidence intervals for each of the performance measures are also presented. Figures 16, 17, and 18 give the 95% confidence limits for a mean predicted value of MPD, BLK, and ALU, respectively. The voice and data arrival rates for these graphs correspond to a fairly heavy traffic load (CS = 4, PS = 400, SERV = 180).

4.4.3 Sensitivity to Network Size

In addition to checking the performance of the simulator for varying traffic loads and link capacities on a fixed network topology, the analysis also examines a fixed traffic load on varying sized topologies. In particular, a throughput requirement of 2000 data packets per second with a voice call arrival rate of 20 calls per minute was imposed upon three different sized networks. A 10-node, 20-node, and 52-node network were each subjected to the fixed traffic load.

The 10-node network is the TYMNET topology shown previously in Figure 4. Six links interconnect the backbone nodes.

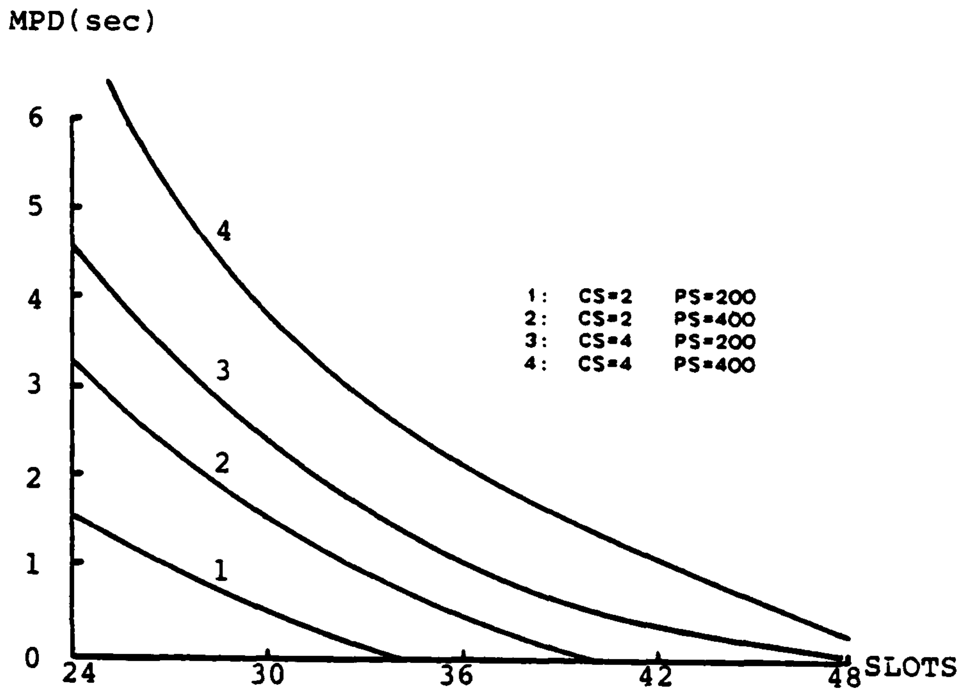


Figure 13. Mean packet delay (MPD) vs. link capacity (SLOTS)

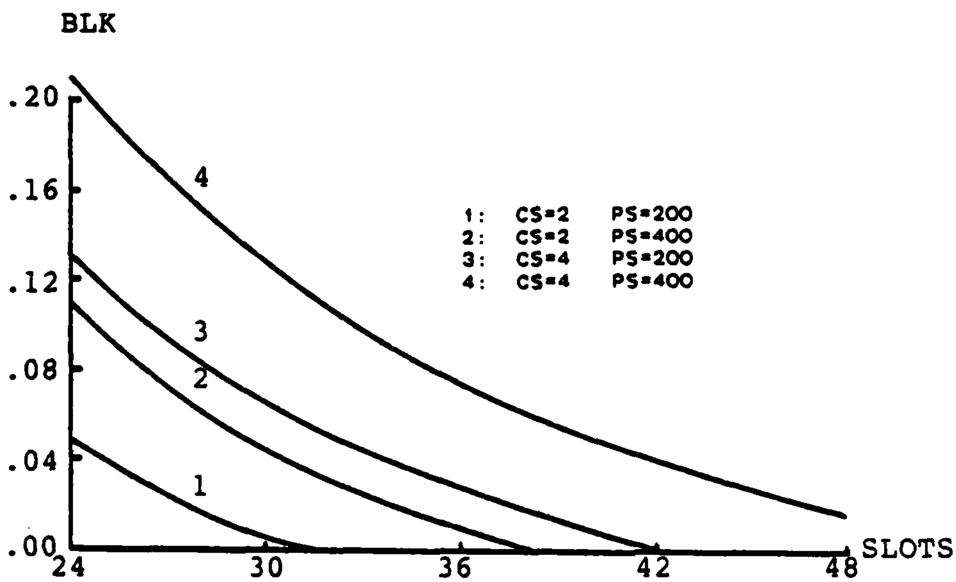


Figure 14. Fraction of calls blocked (BLK) vs. link capacity (SLOTS)

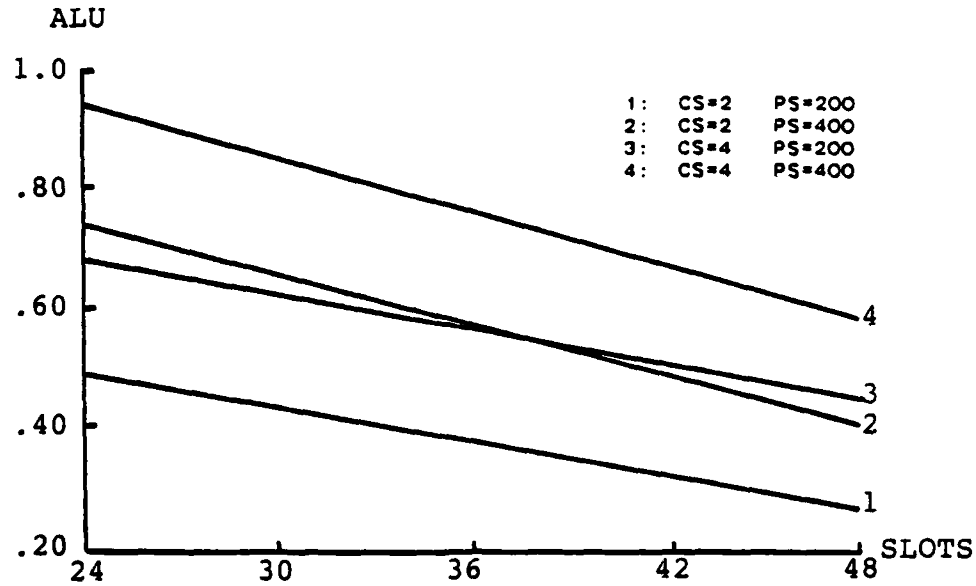


Figure 15. Average link utilization (ALU) vs. link capacity (SLOTS)

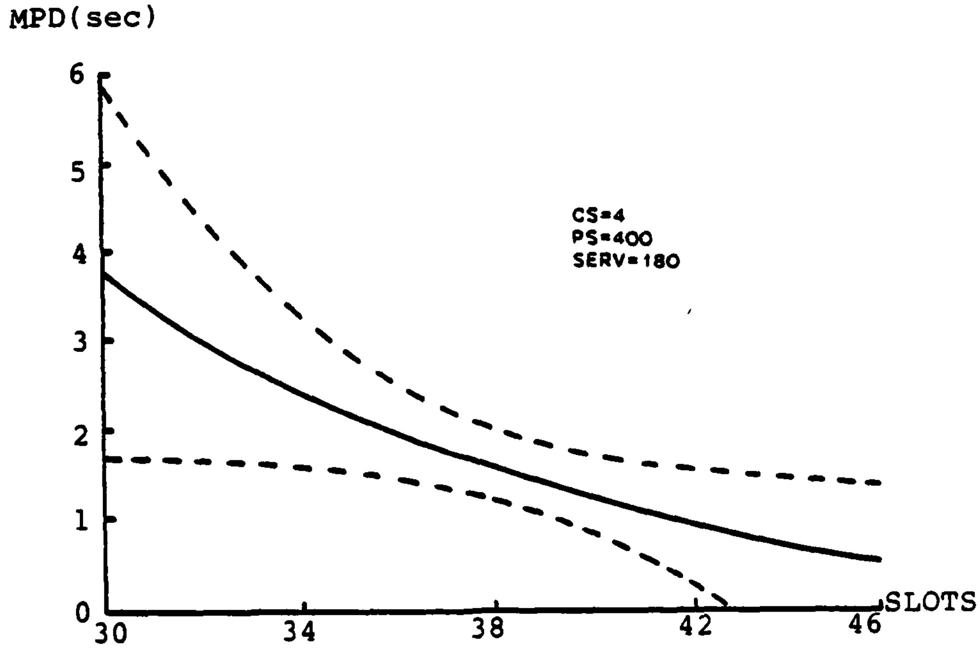


Figure 16. 95% Confidence limits for mean packet delay (MPD)

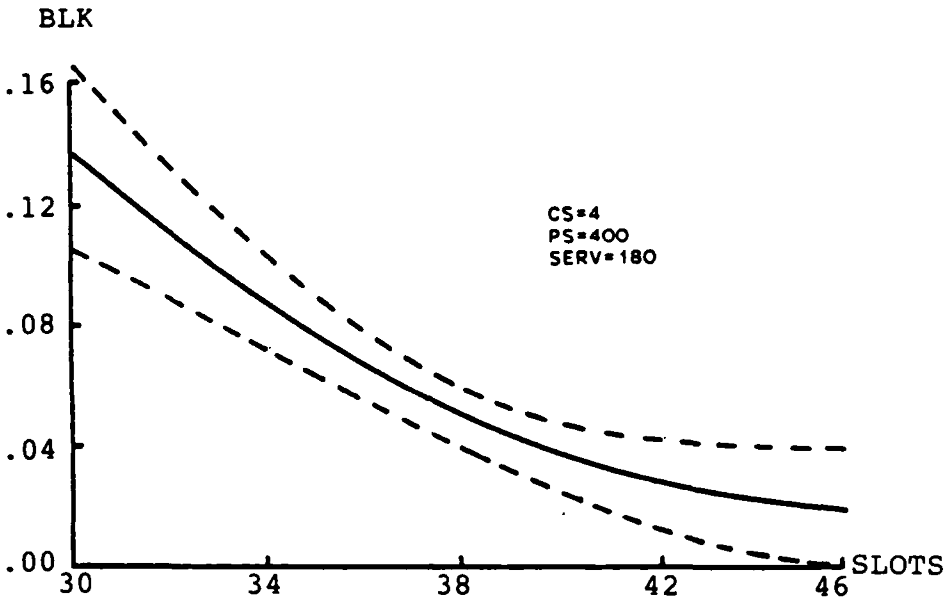


Figure 17. 95% Confidence limits for fraction of calls blocked (BLK)

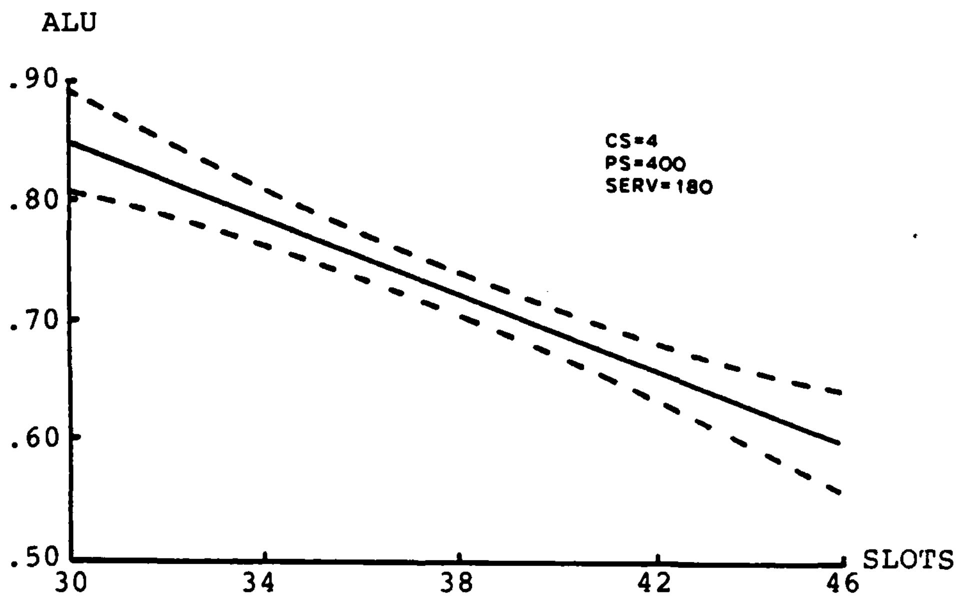


Figure 18. 95% Confidence limits for average link utilization (ALU)

The 20-node network consists of 10 packet switches and 10 circuit switches. The 10 circuit switches forming the backbone of the network are 10 of the major computing centers in the CYBERNET network [74]. The nodes on the subnet are interconnected by 12 trunk lines. The backbone of the 20-node network is depicted in Figure 19.

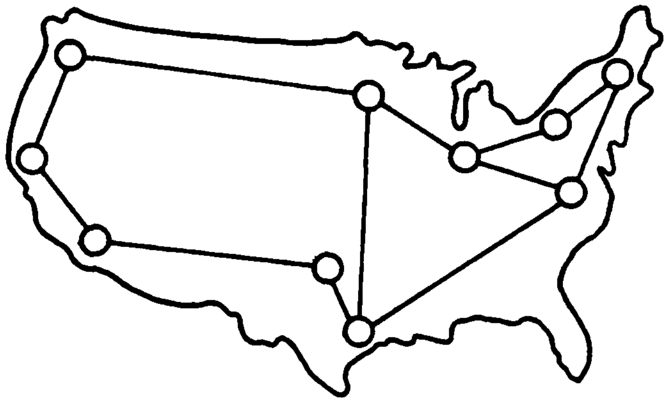


Figure 19. 20-Node network backbone

The 52-node network is comprised of 26 packet switches and 26 circuit switches, with the circuit switch nodes corresponding to a 26-node substructure of the ARPANET network. This 26-node subset of ARPANET is commonly used in the literature for comparative analyses [7, 14, 32, 42]. The subnet is interconnected with 33 links. Figure 20 shows the communications subnet of the 52-node network.

Figure 20. 52-Node network subnet

All links in the three topologies have a fixed capacity of 40 slots. Additionally, each of the three communications subnets is node-biconnected (i.e., there are at least two node-disjoint paths between any pair of nodes).

Table III summarizes the results obtained when subjecting the three different network topologies to the given workload. The performance measures given for the 10-node network are averages obtained from a total of 10 simulation runs. Correspondingly, the data for the 20-node and 52-node networks are averages of 7 simulations

Table III. Sensitivity to network size

Topology	Subnet Link/Node Ratio	MPD	BLK	ALU
10-Node	1.2	.987	.034	.654
20-Node	1.2	.451	.033	.416
52-Node	1.27	.313	.006	.289

and 1 simulation, respectively. Despite the fact that there is a reduction in MPD as the number of nodes increases, the relatively smaller decrease in MPD when going from the 20-node network to the 52-node network (as compared to going from the 10-node network to the 20-node network) results from the fact that the backbone of the integrated network is circuit-switched. This implies that there is a fixed switching delay (assumed to be 50 ms in this study) at each node, and each packet incurs this delay at each intermediate node on its route. Hence, if the subnet link/node ratio remains fairly constant, the MPD is expected to decrease to a certain point and then begin to increase as the number of nodes in the network increases. This phenomenon is due to the fact that as more nodes are added to the network, a greater proportion of the delay can be attributed to switching delays, even though the queueing delay steadily decreases. However, as technology improves the switching delays, the effect of

this phenomenon on total MPD is reduced.

4.5 Summary

Simulation model validation is a never-ending process, but a well designed sensitivity analysis of the simulator can increase the user's confidence in the model as well as his knowledge of the model. In this regard, a sensitivity analysis is an important step in the direction of model validation. This chapter has outlined an approach to analyzing the performance characteristics of an integrated computer-communication network simulation model and has presented the results of the analysis.

The investigation has centered on the integrated network traffic load parameters of packet arrival rate (PS), voice call arrival rate (CS), and voice call service times (SERV), as well as the design parameters of link capacity (SLOTS) and network size. Network performance has been measured in terms of mean packet delay (MPD), a strict data measure; fraction of voice calls blocked (BLK), a pure voice criterion; and average link utilization (ALU), a gauge which tends to combine both the data and voice attributes of an integrated network.

The analysis has determined a range of input parameters for which second-order response surfaces can be used to adequately describe realistic network performance. This range is summarized as follows:

CS: 1-6 calls/min

PS: 100-600 packets/sec

SERV: 150-210 sec

SLOTS: 24-48 slots

In light of the fact that an integrated network with a circuit-switched subnet has not yet been implemented, the term "realistic" performance is admittedly dubious. However, performance criteria for current packet-switched and circuit-switched networks can and have been used as guidelines (e.g., a packet delay of 10 seconds is clearly unacceptable, for the message would automatically time out), although flexibility in the guidelines has been preserved so as to not stifle the range of model applicability.

Furthermore, the graphs presented in this chapter consistently support the theme that the performance measures are more sensitive to the CS (voice) arrival rate than to the other parameters investigated. The "slices" (i.e., graphs) of the response surfaces that correspond to an increased CS level generally have higher ordinate values and steeper slopes than the "slices" that correspond to an increased network loading due to variations in the other parameters. In essence, voice arrival rates tend to dominate the network in the sense that the virtual circuits established by successful call initiations provide the framework of paths by which data

packets can "piggyback" the digitized voice.

Additionally, heavily loaded networks tend to intensify the effect of any parameter. Increasing the number of nodes in a network (along with a corresponding increase in the number of links) for a fixed traffic load will decrease link utilization rates and call blocking, but may or may not decrease mean packet delay, depending on what proportion of the delay is switching delay.

The simulation model analyzed in this research is a tool that can be used by integrated network designers and managers alike. As a result of this analysis, the user of the simulation model now has a more precise understanding of the relationship between integrated network performance and the network parameters that influence such performance. In light of this, the model is a more viable tool now than it was prior to the analysis.

CHAPTER V

DEVELOPMENT OF AN ADAPTIVE TOPOLOGICAL CONFIGURATION MODEL

5. Problem Formulation

The topology design problem for an integrated computer network can be stated as follows [3, 14, 42, 95]:

Given:	packet switch and circuit switch node locations
	data traffic requirement between packet node pairs
	voice traffic requirement between circuit node pairs
	cost/capacity matrix
Minimize:	cost of the integrated network
Subject to:	reliability constraint
	packet delay or throughput constraint
	voice call blocking constraint
	link utilization constraint
Variables:	link placement
	link capacity

Other common formulations of the design problem are the following:

1) minimize mean packet delay given a cost constraint, and
2) maximize throughput given cost and delay

constraints.

However, it has been shown [40, 42] that all of these formulations are closely related and that the solution techniques that apply to the stated formulation above also apply to the other formulations.

The topological design of an integrated network means assigning the links and link capacities that interconnect the circuit-switched, backbone nodes. The nodes, locations, or sites are the sources and sinks of the information flow. The data traffic matrix specifies how many packets per second must be sent between nodes i and j. Similarly, the voice traffic matrix tells how many calls per minute are initiated at node i and directed to node j. The cost/capacity matrix gives the cost per unit distance for each of the various speed transmission links available, as well as the fixed charge for each line type. There are generally only a discrete number of link capacities (speeds) available, e.g., 50 Kbits/sec, 500 Kbits/sec, 1 Mbits/sec.

The specification of constraints establishes a grade of service for the network. The reliability constraint is usually given in terms of k-connectivity. When k = 2, a most common situation [31], the biconnectivity constraint indicates that there must be at least two node-disjoint paths between every pair of nodes. Mean packet delay is a common delay constraint, and it is usually given in terms

of "not to exceed a specified number of milliseconds or seconds". Call blocking and link utilization are usually given in percent or a decimal fraction between 0 and 1.

5.1 An Iterative Approach to Network Design

The goal of any topological design procedure is to achieve a specified performance at minimum cost. The design problem as stated above can be formulated as an integer programming problem, but the number of constraint equations quickly becomes unmanageable for even small problems. In fact, the optimal topological design solution for networks with greater than ten nodes is believed to be computationally prohibitive [14, 40]. This is indeed plausible since Garey and Johnson [39] have shown the network reliability problem, which is a subproblem of the topology design problem, to be NP-hard. A viable alternative to finding the optimal solution is to use a computationally efficient heuristic to generate suboptimal solutions. This is the approach taken in this research.

The technique is to generate a starting topology and evaluate this topology using the simulation model described in Chapter III. The network topology is then perturbed in a manner determined by a heuristic. The heuristic uses the performance data obtained from the simulation to move the topology in the direction of

satisfying the set of constraints. The perturbed topology is once again evaluated via simulation and the heuristic reapplied. The performance feedback mechanism, or heuristic, is applied repeatedly after each simulation until all of the performance constraints are satisfied, if possible. Once all the constraints are satisfied, a "feasible" solution has been obtained. The model continues to try to improve on the feasible solution by repeated use of the heuristic until it can no longer do so, at which time the best feasible solution is considered to be a "local optimum". The iterative approach is depicted in Figure 21 where the heuristic is shown as a performance feedback mechanism tying the three main modules of the model in a loop.

The entire procedure can be repeated with other starting topologies, thereby generating a sequence of local optima. The situation of having many suboptimal solutions rather than the optimal solution is not all that bleak. Many factors usually enter into a design decision and modeling and analysis may be just one of them. The existence of several appropriate solutions could very well increase the flexibility of incorporating other nontechnical factors (e.g., political considerations) into the design decision.

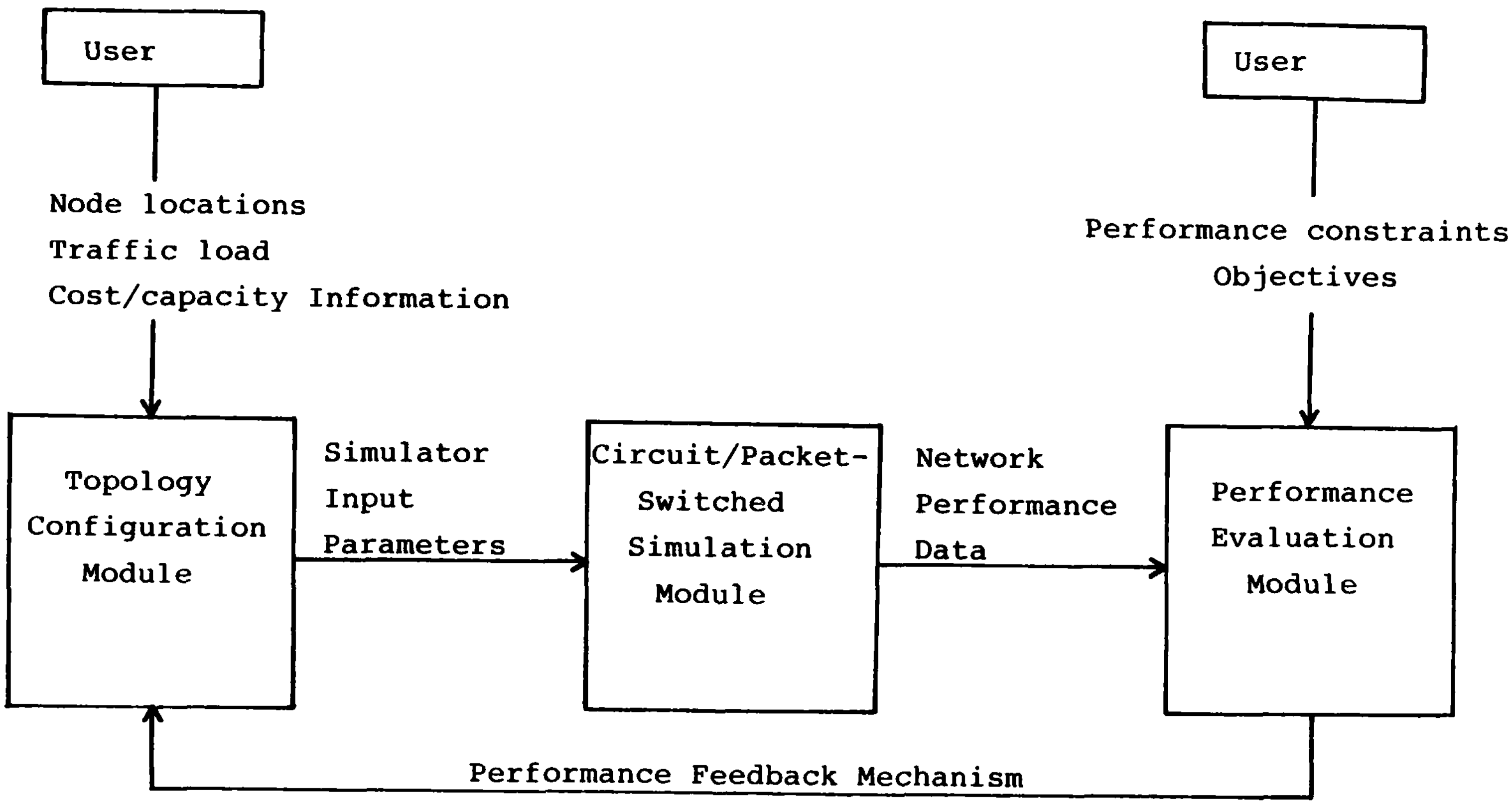


Figure 21. Iterative approach

5.2 Starting Topology Generation

The input to the starting topology generation process is the set of node locations, the traffic requirements, and the cost/capacity information for the set of leased lines available. The output of this process is a set of links with one of a discrete set of link capacities assigned to each existent link. That is, an integer matrix C can be used to describe a topology. If c_{ij} = 0, then there is no link between nodes i and j. Positive integers for c_{ij} indicate the link type or capacity between nodes i and j. Matrix C is referred to as the topology connectivity matrix and is symmetric with respect to the main diagonal, always a set of zeros. The process of generating a starting topology consists of two major steps. Each of these steps is now considered in detail.

5.2.1 Link Assignment Approach

If the link assignment process were such that the resulting initial topology were feasible, many of the subsequent problems involved in heuristic selection could be avoided. The technique implemented in the model is not quite so ambitious and does not guarantee a feasible starting topology. Instead, the approach is aimed at efficiently generating a network that is reasonably close to feasibility but at the same time has a relatively low cost. The algorithm used is a modified version of a

heuristic due to Steiglitz et al. [93]. The heuristic is based on Whitney's theorem [98] which essentially states that if a network topology is k-connected, then every node in the network must have degree of at least k. The degree of a node is the number of links or arcs incident upon that node. Links in a network topology are undirected edges in a graph whose vertices correspond to the nodes in the backbone of the network. Although the condition that each node be of degree k is a necessary condition in a k-connected network, this condition is not sufficient [98].

The approach taken is called a link deficit approach. The difference between the required number (k) of links needed at a node and the actual degree of that node is called the link deficit for that node [95]. The algorithm can be described as follows:

(1) The nodes are randomly numbered. It is the randomization of nodes that renders the algorithm nondeterministic and which allows many starting topologies to be generated from the same input data.

(2) Determine the node with the highest link deficit. Call it A. Ties are broken by the ordering of nodes.

(3) Determine the node with the highest link deficit that is not already linked to node A. Call it

B. Ties are broken by using minimum distance from A as a criterion or by the ordering of nodes in case the distances are the same.

(4) Add link AB to the network and repeat steps 2-4 until all nodes have degree of at least k.

(5) If the network is connected (i.e., every node is capable of communicating with every other node), then stop.

(6) Otherwise, determine the shortest link that spans two different connected components and insert this link. Go to step 5.

Figure 22 shows the result of applying steps 1-4 above to 10 nodes in the CYBERNET network (see Fig. 19) under the assumption that $k = 2$. The numbers on each of the links indicate the order in which the links were added to the network. At this point the network is still not connected. Execution of steps 5 and 6 adds a link between nodes 3 and 6, resulting in a connected topology. Application of the algorithm guarantees a connected, but not necessarily k-connected network.

5.2.2 Discrete Link Capacity Assignment

The selection of link capacities from a finite set of options is one of the most difficult of all network design problems [32, 42, 45]. Furthermore, because digitized voice and packet data are being superimposed on a common

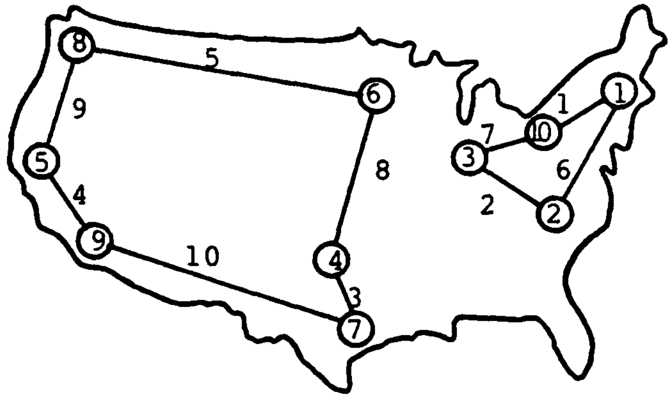


Figure 22. Link deficit approach to assigning links

carrier, the capacity assignment problem is even more complex in an integrated network than in either the circuit-switched or packet-switched network. There are no closed form solutions even in the case where link cost is a linear function of channel capacity [45], hardly a realistic situation [21, 25, 42, 52, 95].

Complicating matters is the fact that the capacity assignment problem is intimately related to the routing problem. In order to properly assign link capacities, an estimate of the traffic load on each link is needed. But the traffic on a link is highly dependent on the routing scheme used. Extensive research has been conducted on the design and analysis of routing algorithms [6, 36, 43, 64, 65], and the literature abounds with routing classification schemes [48]. One simple classification of routing schemes is found in Tanenbaum [95] who categorizes routing algorithms as either static (fixed) or

dynamic (adaptive). Dynamic algorithms base their routing decisions on the current load, so consequently it is difficult to estimate traffic loads when adaptive routing is used in a network. The possibility of designing a network using one routing algorithm and operating the resulting network with another routing algorithm is real and could result in poor performance. Fortunately, it has been shown [9, 40] that the performance of fixed, multiple-path routing approximates that of optimal adaptive routing under stable conditions, i.e., where the traffic load in a network is not concentrated between a small percentage of the nodes.

The approach taken to assign link capacities in an integrated network is based on the shortest distance criterion, a commonly used strategy in the literature [20]. An outline of the procedure is as follows:

(1) For each pair of nodes, A and B, find the shortest path between nodes A and B and label the path as $X_1X_2\ldots X_n$, where $A = X_1$, $B = X_n$, and $X_2, X_3, \ldots, X_{n-1}$ are the intermediate nodes on the shortest path between A and B. Floyd's algorithm [24, 72] is used to determine the shortest path between each pair of nodes.

(2) For a given node pair A and B, add the packet traffic load of (A, B) or (B, A), whichever is

greater, to each link on the path from A to B. Similarly, add the voice traffic load of (A, B) or (B, A), whichever is greater, to each link on the path. A voice digitization rate of 32 Kbits/sec is used to transform voice arrival rate units to link capacity units (bps).

(3) Repeat step 2 for each node pair.

(4) For each link in the network, assign the smallest discrete link capacity that is greater than or equal to the estimated integrated traffic load determined in step 2. If there is no such option available, then assign the largest available link capacity.

Suppose it is assumed that each of the 10 circuit switch nodes in Figure 22 has an average voice call arrival rate of 4 calls per minute and that associated with each circuit switch node is a packet switch node having a data packet arrival rate of 400 packets (1000 bits/packet) per second. Application of the above heuristic under the assumed uniformly distributed network traffic load results in the starting topology shown in Table IV. The topology is given as a connectivity matrix. This table also shows the cost and capacity information for the five options as well as the coordinates of the randomized nodes in the backbone. Under these conditions the cost of this starting topology is $2112.39 (a monthly

charge).

Table IV. Starting topology connectivity matrix

LINE CAPACITY AND COST INFORMATION:

LINE TYPE	CAPACITY (BPS)	COST($) PER UNIT LENGTH	FIXED COST($)
1	800000.	1.00	50.00
2	1000000.	2.00	50.00
3	1200000.	4.00	100.00
4	1600000.	8.00	150.00
5	2000000.	16.00	200.00

THE RANDOMIZED NODES:

I	X(I)	Y(I)
1	25.50	13.80
2	23.10	10.90
3	19.40	10.50
4	13.40	6.10
5	1.10	11.10
6	15.10	13.90
7	14.30	4.20
8	2.80	17.20
9	2.40	8.30
10	24.30	12.40

THE CONECT MATRIX NOW LOOKS LIKE:

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	0	0	0	0	4
2	1	0	2	0	0	0	0	0	0	0
3	0	2	0	0	0	5	0	0	0	5
4	0	0	0	0	0	4	3	0	0	0
5	0	0	0	0	0	0	0	5	4	0
6	0	0	5	4	0	0	0	5	0	0
7	0	0	0	3	0	0	0	0	1	0
8	0	0	0	0	5	5	0	0	0	0
9	0	0	0	0	4	0	1	0	0	0
10	4	0	5	0	0	0	0	0	0	0

THE COST($) OF THIS TOPOLOGY IS: 2112.39

5.3 Network Topology Optimization

Once a starting topology has been generated and the corresponding network performance and cost ascertained, it remains to determine whether or not modifications to the topology can enhance performance or decrease the cost or both. The procedure used to modify a topology with the expectation of improving performance and/or decreasing

cost stands as the crux of any iterative approach to network design. Most of the techniques seen in the literature are geared to localized transformations or minor modifications that hopefully progress in a stepwise fashion to a local minimum.

5.3.1 Perturbation Techniques

Several sources in the literature [3, 14, 20, 38, 42, 44, 58, 59, 95] provide comprehensive surveys of heuristic algorithms that have been or are being used in network design. Three of these heuristics stand out as milestone approaches to perturbing a topology, that is, modifying the number of links and/or the capacity of links. These three approaches, all of which have been applied almost strictly to either packet switching or circuit switching networks, are described here to serve as a background for the approach taken in this research.

5.3.1.1 Branch Exchange Method [14, 42, 44, 93, 95]

The branch exchange (BXC) method seeks an improved design by adding links which are adjacent to deleted links. Two links are said to be adjacent if they share a common node. Possible criteria for choosing the links to be deleted or added are link utilization rates, cost, and estimated traffic loads between node pairs. Figure 23 illustrates two possible exchanges from a given starting

topology. Figure 23 (a) is the starting network with links AD and EF assumed to be the links selected for deletion. Figures 23 (b) and (c) show two possible topologies that could result from the deletion of links AD and EF.

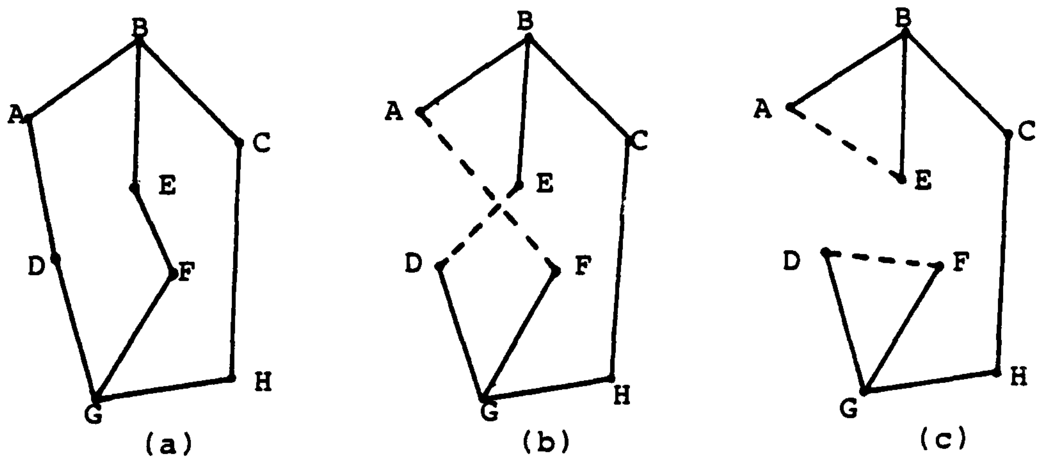


Figure 23. Branch exchange heuristic: (a) starting topology, (b) and (c) resultant topologies

5.3.1.2 Concave Branch Elimination Method [14, 40, 42, 101]

The concave branch elimination (CBE) method which was developed by Gerla [40] starts with a fully connected network and eliminates links until a local minimum is reached. A fully connected topology is one in which each node is connected directly to every other node. That is, if a network has N nodes, then the fully connected topology has N(N-1)/2 links. The scope of applicability

of the CBE method is limited, however, to cases where the discrete costs can be reasonably approximated by concave functions and the packet queueing delay can be adequately described by the Pollaczek-Khintchine formula [14, 20]. This formula is a concise analytical expression which gives the average queueing delay for a single-server system having Poisson arrivals and an arbitrary distribution of service times [20]. An example of link costs that can be approximated by concave functions is given in Figure 24 [42].

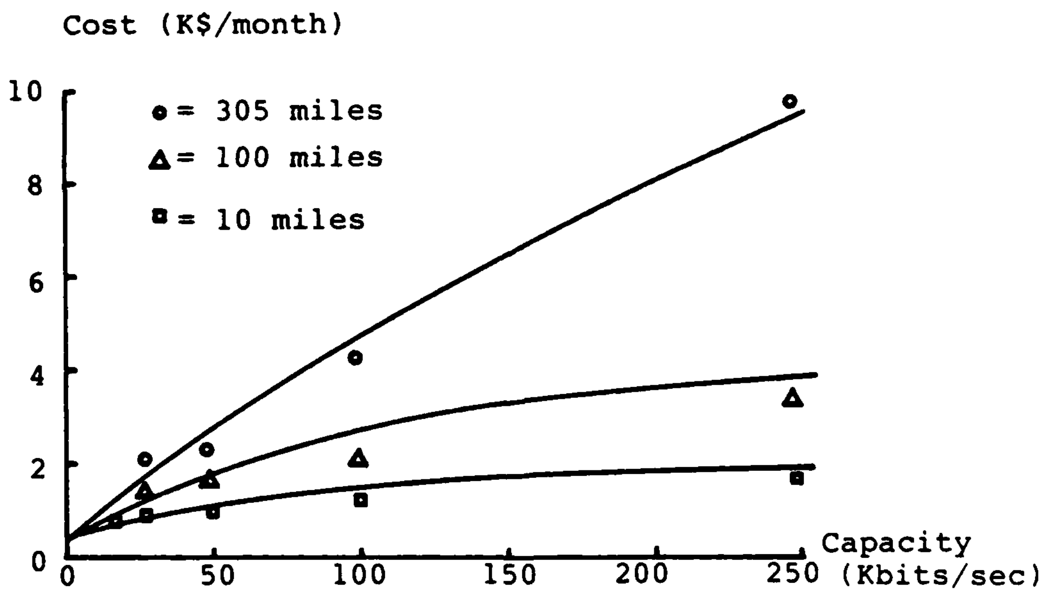


Figure 24. Concave approximations to link costs for various link lengths

The CBE method is computationally a more efficient design procedure than the BXC method [14], but its

applicability is extremely limited. The CBE heuristic is mentioned here not only because it represents a significant departure from the BXC process but also because its concept of approximating discrete costs with concave functions is used to determine bounds on heuristic performance. A complete discussion on the techniques of developing lower bounds on the cost of the optimal solution is presented in Gerla et al. [42, 44]. The existence of such bounds allows the appraisal of a heuristic algorithm's accuracy.

5.3.1.3 Cut-Saturation Method [3, 14, 42, 44, 95]

Both the BXC and CBE methods possess inherent deficiencies. The BXC heuristic requires an exhaustive search of all local transformations and is very time consuming when more than 20 nodes are involved. The CBE algorithm, although it can efficiently eliminate uneconomical links, does not have a link insertion capability. So once a link is deleted there is no possibility of recovering that link. As a result of these deficiencies, a new method, the cut-saturation (CS) algorithm, evolved.

The cut-saturation method can be viewed as an improved BXC algorithm. Rather than performing all possible link exchanges as the BXC method does, the CS algorithm selects only those exchanges that are likely to

improve delay and cost. A saturated cut (or cutset) in a network is defined [42] to be the minimum set of most utilized links that, if removed, leaves the network partitioned into two disjoint components of nodes. If the links in the network shown in Figure 25 are numbered in the order from most utilized link to least utilized link, then the cutset is seen to be the set {1, 3, 5}.

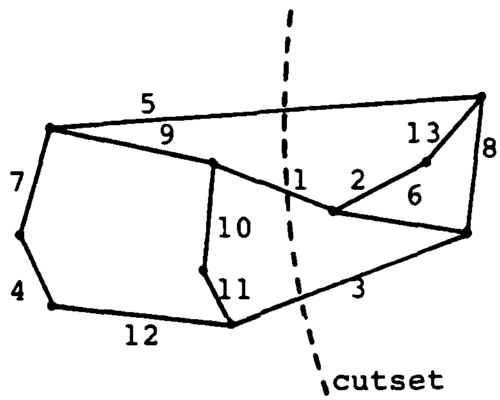


Figure 25. Example of a saturated cut

Tanenbaum [95] gives a scheme to find the cutset. First, rank order the links from most utilized to least utilized. Then remove from the ordered list one link at a time until the network is split into two disjoint connected components. In Figure 25, this occurs after links 1, 2, 3, 4, and 5 have been removed. To find the minimum cut put each of these links back into the network in turn. If putting a link back into the network does not

reconnect the network, then that link is not part of the cutset. Such is the case in Figure 25 for links 2 and 4. Hence, the cutset does not include these links.

The saturated cut in a network functions somewhat like a bridge between the two components. In fact, the capacity of the cutset bounds the throughput that a network could possibly realize. Hence, it seems reasonable that if a link is to be added to improve throughput or delay, that link ought to span the cutset by joining the two disjoint components. Various criteria exist as to which nodes in the two components should be connected. A commonly used criterion is to add the link having the lowest cost, which also may be the shortest link, depending on the tariff structure. Similarly, link deletions should occur only within each of the individual components. Link utilization and cost usually determine the link to be deleted. While minor variations of the cut-saturation algorithm exist [14], the substance of these algorithms remains as that described above.

A comparative analysis of the three heuristics presented here has shown that the CS algorithm gives better results and is computationally more efficient than either the BXC or CBE methods [14, 42, 44]. Additionally, a comparison of CS solutions to theoretical lower bounds shows that the CS algorithm can produce near-optimal solutions [44].

5.3.2 A Two-Phase Approach to Network Optimization

The selection of an optimization heuristic can depend on a variety of factors. Among them are the cost-capacity relationship, the topological constraints involved, the desired accuracy, the degree of human interaction, and the type of network being designed. The three perturbation techniques described above have resulted from and been applied to primarily packet-switched networks. The design and analysis of integrated or hybrid networks is in its infant stage, and the literature is almost void of any kind of technique that is specially suited for optimizing integrated networks. Gitman et al. [45] state that there are basically two approaches to the integrated network design problem:

(1) solve the link/capacity problem for the voice traffic and data traffic separately, or

(2) solve the link/capacity problem for the combined voice and data traffic.

Gitman's approach has been to use option 1 together with a CBE heuristic [45]. Option 2 is the approach taken in this research.

A two-phase approach to the optimization of integrated networks has been adopted in this research. Phase 1 concerns itself with finding a feasible solution, while phase 2 attempts to improve performance and cost

while maintaining feasibility. In phase 1, the topology is modified in a manner designed to satisfy the reliability, blocking, and delay constraints. This involves using a heuristic that will in general add capacity to the network and consequently increase the cost. Phase 2 attempts to modify the topology in a manner that will decrease the cost of the network while still satisfying the three constraints. This phase involves the use of a heuristic that will in general decrease the total capacity of the network. Decreasing capacity means decreasing the cost and increasing the utilization, since capacity and utilization are inversely related. The entire process can be depicted as a cubic polynomial as shown in Figure 26, where positive slopes are associated with phase 1 and negative slopes with phase 2. A description of the heuristic used in each phase is now presented.

5.3.2.1 An Integrated Cut-Saturation Add Heuristic

The add heuristic used in the adaptive configuration model is based on the cut-saturation algorithm. If a link is to be added, the nodes which form the link are determined by assigning two weights to each node on the backbone of the network. One weight is for the voice call blocking activity at the node and the other weight is for the data packet delay incurred at that node. The weights

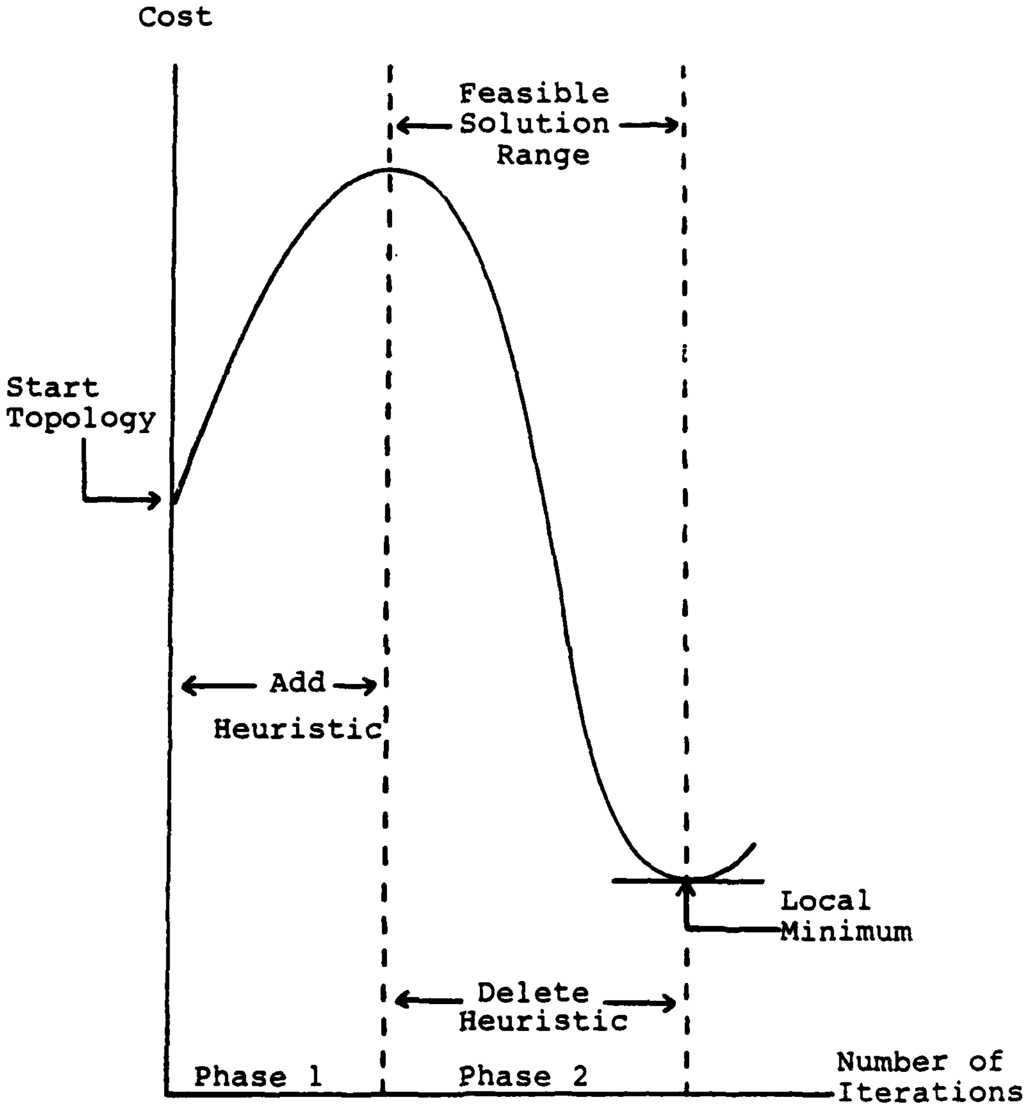


Figure 26. Two-phase approach to topology optimization

are the number of standard deviations, or z-values, that the individual node statistic (blocking or delay) deviates from the network mean statistic. If neither the delay nor blocking constraint is satisfied, then the final node weight is found by summing the two individual weights. If exactly one of these constraints is not satisfied, then only the weight of that particular statistic is taken as the final node weight. The link added is the one for which the combined weights across the cut is greatest.

The add heuristic may on any one iteration add more capacity to one or more links, add one or two new links, or it may do both. An outline of the heuristic's action is as follows:

Let MIN = minimum average link utilization constraint,

DELTA = a decimal fraction (e.g., .10) which is an indication of the heuristic's step size,

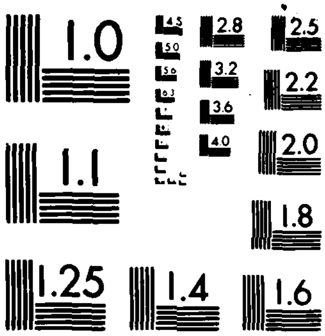
UTIL(i) = the actual utilization of link i,

NCAPS = number of discrete capacity options,

C(i) = current capacity of link i, where C(i) is an integer variable that can take on the values 0, 1, 2,..., NCAPS,

D, B, R = indicator variables for the delay, blocking, and reliability constraints,

AD-A141 309 ADAPTIVE TOPOLOGICAL CONFIGURATION OF AN INTEGRATED 2/3
CIRCUIT/PACKET-SWITCHED COMPUTER NETWORK(U) AIR FORCE
INST OF TECH WRIGHT-PATTERSON AFB OH M J KIEMELE 1984
UNCLASSIFIED AFIT/CI/NR-84-16D F/G 17/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

respectively, where a 1 means the constraint is satisfied and a 0 means it is not satisfied, and

ADD = an indicator variable, where if ADD = 1 a link will have to be added (initially, ADD = 0).

(1) If D = 1 and B = 1, go to step (4).

(2) For each link i such that UTIL(i) > MIN, calculate $N(i) = \lceil (UTIL(i) - MIN)/DELTA \rceil$, where $\lceil X \rceil$ denotes the smallest integer greater than or equal to X. $N(i) \geq 0$ for each i. If N(i) = 0 for all i, then set ADD = 1.

(3) For each link i, modify C(i) to C(i) = C(i) + N(i). If C(i) > NCAPS, then set C(i) = NCAPS and ADD = 1.

(4) If R = 1 and ADD = 0, then no link will have to be added, so stop.

(5) If ADD = 1, then calculate the cutset and determine the components on either side of the cut.

(6) If R = 1 and ADD = 1, then add the link determined by the weighting scheme described above and stop.

(7) If R = 0 and ADD = 0, then a link need be added only to satisfy the reliability criterion. Biconnectedness (k = 2) is the

assumed reliability constraint in the model, and an algorithm to determine the biconnected components has been implemented [2]. The link chosen to be added is the least costly link that spans two of the biconnected components. Add the link and stop.

(8) If R = 0 and ADD = 1, then at least one link is added. The link chosen is the link with highest weighting across the cut that also spans two of the biconnected components, if such a link exists. If no such link exists, then two links are added. Specifically, the links selected are the ones that would be selected in steps (6) and (7). Add the link(s) and stop.

5.3.2.2 A Biconnectivity-Preserving Delete Heuristic

Phase 2 of the optimization process inherits a feasible topology and seeks to reduce the cost of the network while preserving feasibility. The cost reduction, with a corresponding increase in utilization, is attained by decreasing the total capacity of the network. A biconnectivity-preserving delete heuristic is used in this research to systematically move toward a local optimum. Topological perturbations in phase 2 are always performed on a feasible topology, and deleting a link that would reduce connectivity to something less than biconnectivity

(the reliability criterion) is prohibited. The heuristic is applied to the "best" feasible topology obtained up to that point in the iterative process. This necessitates the storing of the "best" topology, as well as keeping a record of unsuccessful perturbations on this topology. A limit is placed on the number of consecutive perturbation failures allowed, and when this limit is reached, the "best" topology is taken as a local optimum.

The heuristic selects a link and determines whether or not the link is to be deleted. If it is not to be deleted, it then determines by how much capacity the link should be reduced. A description of the heuristic's action is now presented. The notation is the same as in the previous section.

(1) Find the least utilized link of those still qualified for investigation. Call it z.

(2) If C(z) = 1, go to step (5).

(3) If UTIL(z) $\leq$ MIN, then calculate N = $\lceil$(MIN - UTIL(z))/DELTA$\rceil$. If UTIL(z) > MIN, then set N = 1.

(4) Reduce the capacity of link z by N units or to 1, whichever is greatest. That is, set C(z) = Maximum {C(z) - N, 1}. Remove z from the qualified list and stop.

(5) Check to see if link z can be deleted without violating the reliability constraint. If it

can, then delete z, remove z from the qualified list and stop. If it cannot be deleted, then remove it from the qualified list and return to step (1).

5.4 Description of the Model

The development of this adaptive topological configuration model (CIRPAC) for an integrated circuit/packet-switched computer network was based on a top-down design and a bottom-up test and integration procedure. The code was written in FORTRAN and the development and analysis of the model has been performed on the Amdahl 470. A listing of the code which is functionally self-documented is given in Appendix A. Appendix B provides additional documentation by describing the variables and giving the dimension requirements for arrays. Appendixes D and E illustrate model input and output, respectively. This section describes the higher-level logical and functional aspects of the model and documents some of the timing and spatial requirements for various conditions.

5.4.1 Logical Description

A high-level logic flow of the entire design optimization process in CIRPAC is shown in the flow diagram of Figure 27. The upper loop corresponds to phase

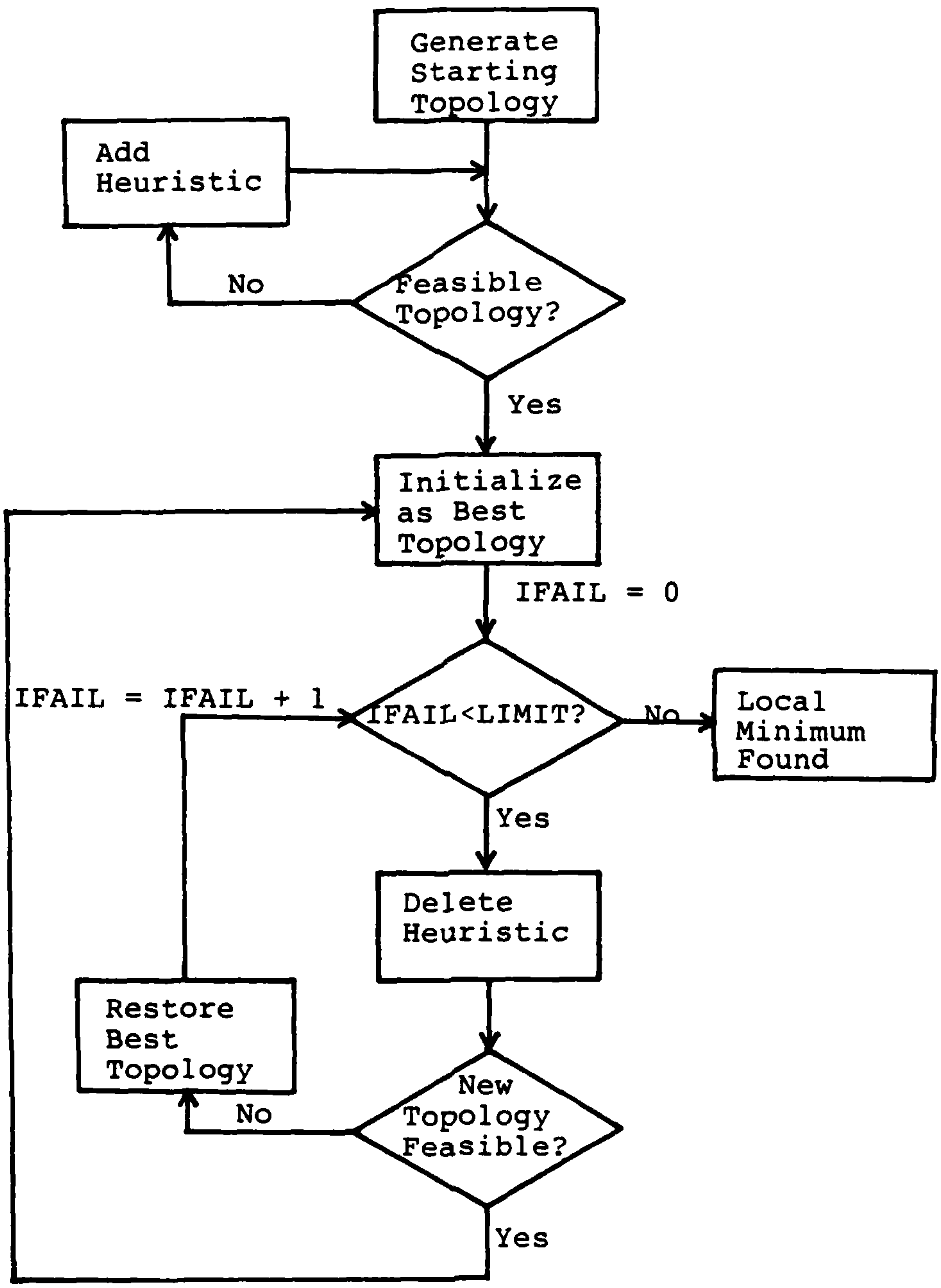


Figure 27. Optimization logic flow diagram

1 and the lower loops make up phase 2 of the optimization approach discussed in the previous section. In order to determine network feasibility, the performance characteristics of the network must be obtained. CIRPAC does this by using the simulator described in Chapter III. The implementation of this logic in CIRPAC was accomplished by the use of six functional modules.

5.4.2 Modular Description

The six major modules of CIRPAC are as follows:

(1) Initialization Module (INITAL)

(2) Topology Configuration Module (TCM)

(3) Interface Module 1 (INFACE)

(4) Integrated Network Performance Generation Module (SIMULA)

(5) Interface Module 2 (OUTFAC)

(6) Performance Evaluation Module (PERFRM)

The names of each of these modules correspond to subroutines which are considered to be the drivers of the respective modules. Each driver subroutine may call other subroutines which aid in accomplishing the module's function. Altogether there are 35 routines including CIRPAC, which is considered to be the controller of the six functional modules. CIRPAC's routine calling hierarchy is shown in Figure 28. A functional description of each of the six modules follows.

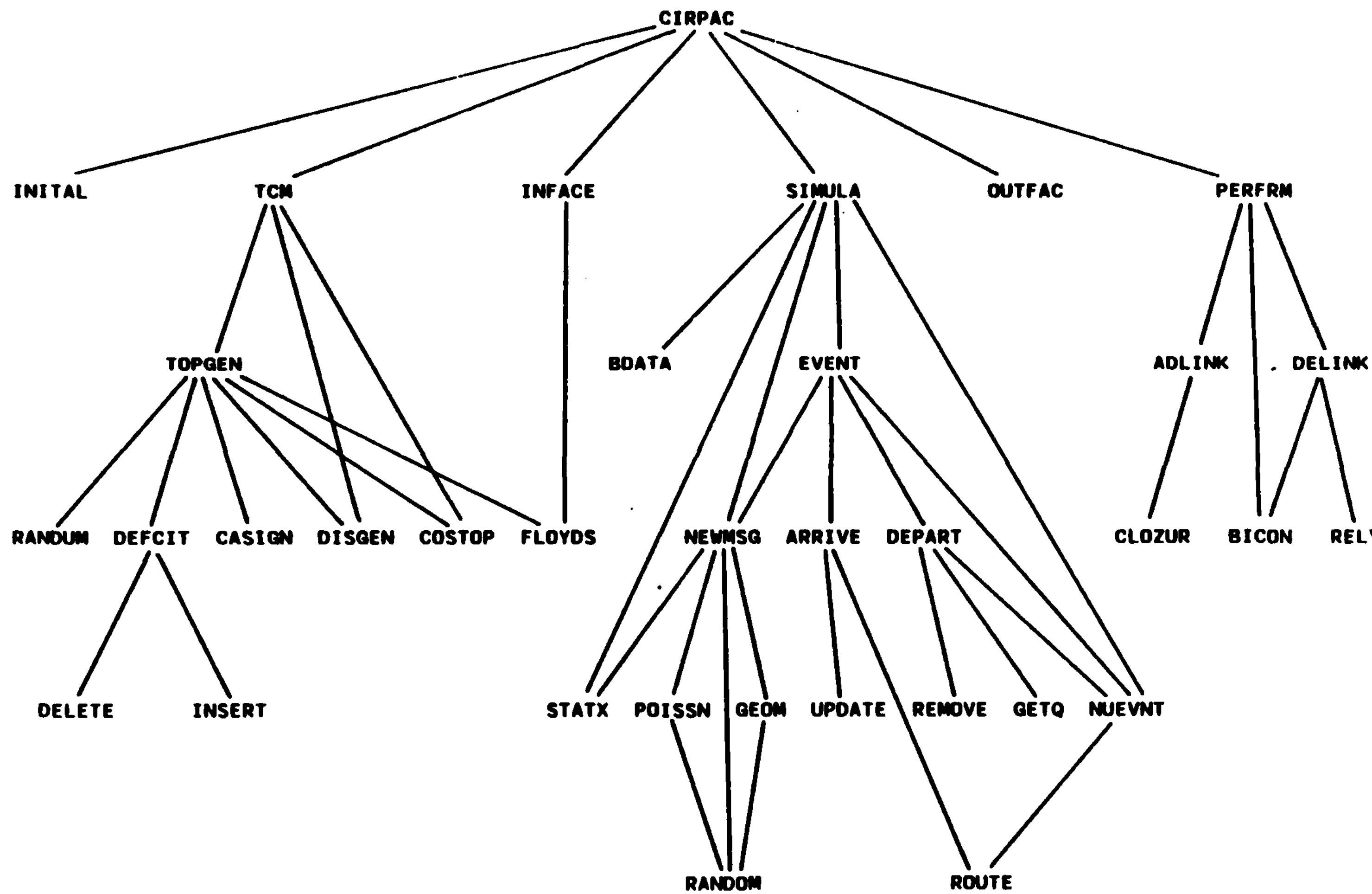


Figure 28. Routine calling hierarchy

5.4.2.1 Initialization Module (INITAL)

The initialization module (INITAL) creates the problem domain. That is, the node or site locations, the traffic workload, the capacity/cost information, and all of the constraints as given in the problem formulation are established. It also supplies the seeds for the random number generators and some control parameters that allow the user more flexibility in applying the model. For example, the user can supply a starting topology rather than having CIRPAC automatically generate one.

5.4.2.2 Topology Configuration Module (TCM)

The primary function of the TCM is to generate a starting topology. The TCM is capable of interpreting user-provided control parameters and directing the logic flow therefrom. Given the direction in which the topology is to move, the TCM actually modifies the topology and prints out the new topology and its associated cost at the start of each iteration. All of the TCM's functions occur within the problem domain.

5.4.2.3 Interface Module 1 (INFACE)

The function of INFACE is to transform the current problem domain into the performance generation domain. That is, it is the interface between the TCM and the network performance generation device which, in CIRPAC, is

the simulator described in Chapter III. The purpose of distinguishing between the problem domain and the performance generation domain is to clearly divide the processes of optimizing the topology and generating network performance data. It provides for a more modularized model design. For example, if some sort of manageable analytical procedure for providing performance data for integrated networks should appear in the future, this device could be used as the integrated network performance generation module in place of the simulator. In that case, only the interface modules would need modification to preserve the CIRPAC design tool.

INFACE is executed prior to each simulation of a topology, for the simulator requires its input data to be in a specific form. It initializes all seed tables and determines all simulator parameters that depend on the topology. Additionally, INFACE uses the topology data to construct the routing tables used by the simulator. The primary routes are based on the shortest distance criterion, and the alternate routes are based on the minimum number of hops criterion. INFACE also checks to be sure that the dimensioning capability of the simulator is sufficient to handle the topology it is to simulate.

5.4.2.4 Integrated Network Performance Generation Module (SIMULA)

SIMULA is the simulation model described in Chapter III, and it generates network performance data via simulation. Generating network performance of a topology means to determine the performance measures of that network. SIMULA generates performance measures for voice, data, and combined voice and data. It operates totally in the performance generation domain simply because it is the performance data generator.

5.4.2.5 Interface Module 2 (OUTFAC)

The function of OUTFAC is to transform the network performance generation domain back into the problem domain. That is, it is the interface (on the output side of the simulator) between the network performance generation domain and the problem domain. Its function is to express the performance statistics generated in SIMULA in a form compatible with the problem domain heuristics that will use the performance data.

5.4.2.6 Performance Evaluation Module (PERFRM)

The performance evaluation module (PERFRM) has the function of determining how a given topology should be modified in order to improve performance and/or decrease cost. The he ristics which comprise the optimization techniqu are an integral part of this module. PERFRM has

the responsibility for remembering the "best" topology as well as those perturbations which have either failed or are no longer eligible for application. PERFRM, which operates entirely in the problem domain, also decides when a local optimum has been found. This completes the descriptions of the six functional modules of CIRPAC. The flow of control (as issued by CIRPAC) between the six modules is shown in Figure 29.

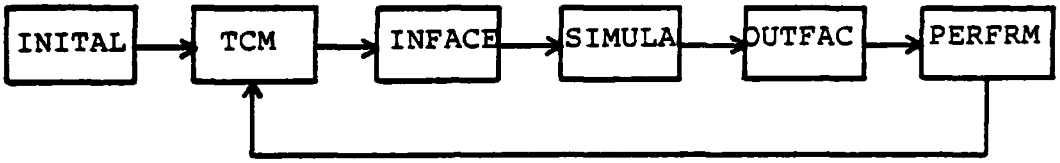


Figure 29. Control flow between modules

5.4.3 Operating Characteristics

Timing and core usage data for the model have been collected for three different sizes of integrated networks. The data presented here are based on a FORTRAN H Extended, optimization level 2, compilation of the source code. Compilation requires approximately 20 seconds of CPU time on the Amdahl 470/V6. Table V summarizes the time and space requirements of the model for three different network sizes when an identical traffic load was imposed on all three networks. The CPU time required is given in CPU seconds needed for each

minute of simulated time. The space data is the compiled object program length in bytes when the dimensioning in the model matches the network size exactly. The packet switch queues in each case contained space for 300 data transactions.

Table V. Model timing and space requirements

Network Size	CPU time/min. of simulation	Space
10-node	5 sec	150K bytes
20-node	9 sec	270K bytes
52-node	30 sec	730K bytes

CHAPTER VI

APPLICATION OF THE ADAPTIVE MODEL

6. Introduction

This chapter presents the results of the application of the automated methodology described in Chapter V to several integrated networks. Integrated circuit/packet-switched networks of 20 and 52 nodes, respectively, have been investigated. Unfortunately, empirical data for integrated networks is at this time nonexistent. However, even though this design tool is intended for integrated networks, the model has been applied to a 26-node packet-switched network that is commonly used in the literature for comparative purposes [7, 14, 32, 42]. The results of applying CIRPAC to this 26-node network are compared with results in the literature.

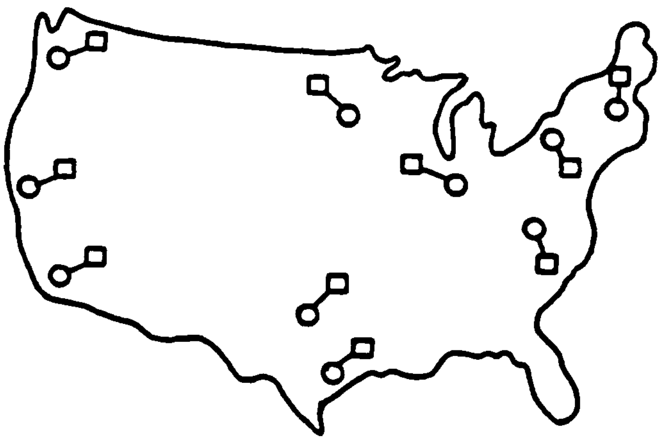
6.1 Application to a 20-Node Integrated Network

CIRPAC has been applied to a 20-node integrated circuit/packet-switched network to obtain 10 different local minima. The network configuration and an analysis of the results are presented.

6.1.1 20-Node Network Configuration

The network analyzed consists of 10 backbone circuit

switches and 10 peripheral packet switches where each circuit switch represents the subnet entry point for packets from exactly one packet switch. The locations of the backbone nodes are the 10 major locations of the CYBERNET and are shown as the circular nodes in Figure 30. The square nodes are the collocated packet switches. The workload assumed consists of a voice call arrival rate of 4 calls/minute at each circuit switch and a data packet arrival rate of 400 packets/second at each packet switch.



LINE CAPACITY AND COST INFORMATION:

LINE TYPE	CAPACITY (BPS)	COST($) PER UNIT LENGTH	FIXED COST($)
1	800000.	1.00	50.00
2	1000000.	2.00	50.00
3	1200000.	4.00	100.00
4	1600000.	8.00	150.00
5	2000000.	16.00	200.00

Figure 30. 20-Node configuration

The traffic is assumed to be symmetric and uniformly distributed between node pairs. The service time for calls is assumed to be exponentially distributed with a mean service time of 180 seconds, and the capacity/cost information is as shown in Figure 30.

6.1.2 Analysis of 20-Node Network

Using 10 different starting topology generator seeds, CIRPAC produced 10 different local minima for the network input data given in the previous section. The results of these 10 cases are summarized in tabular form in Table VI. The "start cost" is the cost of the starting topology and the "best cost" is the cost associated with the local minimum. The delay, blocking, and utilization constraints were as shown in the table. Throughput is defined to be the average number of packets/second traveling out of each node. The total number of iterations is the number of iterations needed to reach the local minimum. Biconnectivity is the assumed reliability constraint.

The local minima obtained can be used to examine integrated network design tradeoffs. For example, if each of the local minima is plotted on a utilization versus cost coordinate system, as shown in Figure 31, it is seen that topology number 3 dominates all of the other nine topologies. That is, each of the other minima has a higher cost and a lower utilization rate than case 3. The

Table VI. 10 Local minima for 20-node network

		AT LOCAL MINIMUM					
Case No.	Start Cost	Best Cost	$\leq$ 1.0 sec Delay	$\leq$ .10 Blocking	> .60 Utilization	Throughput (Packets/Sec)	No. of Iterations
1	2125.02	1925.17	.982	.083	.707	2670	15
2	2472.66	2560.29	.838	.072	.675	3000	19
3	2112.39	1882.81	.963	.085	.740	2730	14
4	2268.81	2150.61	.996	.048	.718	2756	11
5	2774.31	2160.32	.865	.080	.667	2797	14
6	2335.16	2016.40	.999	.088	.706	2639	14
7	2322.21	2386.25	.961	.080	.716	2830	14
8	2790.64	2710.77	.848	.069	.715	2837	16
9	2361.61	2175.87	.984	.083	.698	2785	16
10	2120.75	2307.08	.977	.061	.716	2811	11

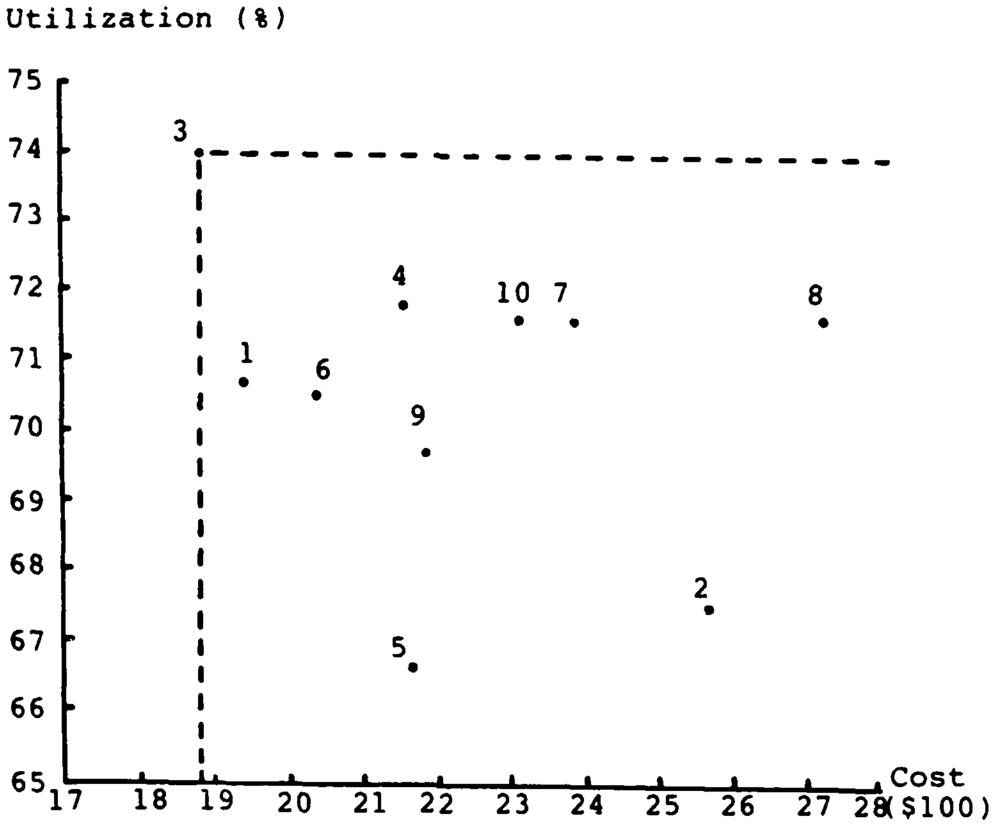


Figure 31. Domination of the utilization/cost space by case 3

topology associated with local minimum number 3 is shown in Figure 32. The integers on each link indicate the line type for that connection. Solid lines represent links that were part of the starting topology while dashed lines indicate links that were added during the optimization process.

On the other hand, if the throughput/cost tradeoff is examined, the plot in Figure 33 results. Case 3 is again a dominator, but this time it dominates only cases 1 and

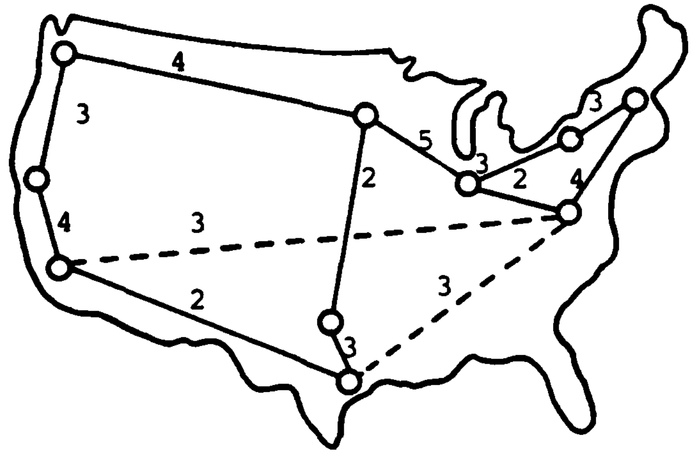


Figure 32. Topology for local minimum #3

6. Additionally, case 2 dominates case 8. If all of the non-dominated points are connected, the plot is capable of telling how much more throughput can be obtained for a given increase in cost. Similar investigations with other combinations of network performance measures are possible and provide the network designer with a degree of flexibility, even though the global optimum is not known.

6.2 Application to a 52-Node Integrated Network

CIRPAC has also been used to optimize a 52-node integrated circuit/packet-switched network. In this case only one local optimum has been generated. The following paragraphs describe the starting topology configuration and track the optimization process until a local minimum

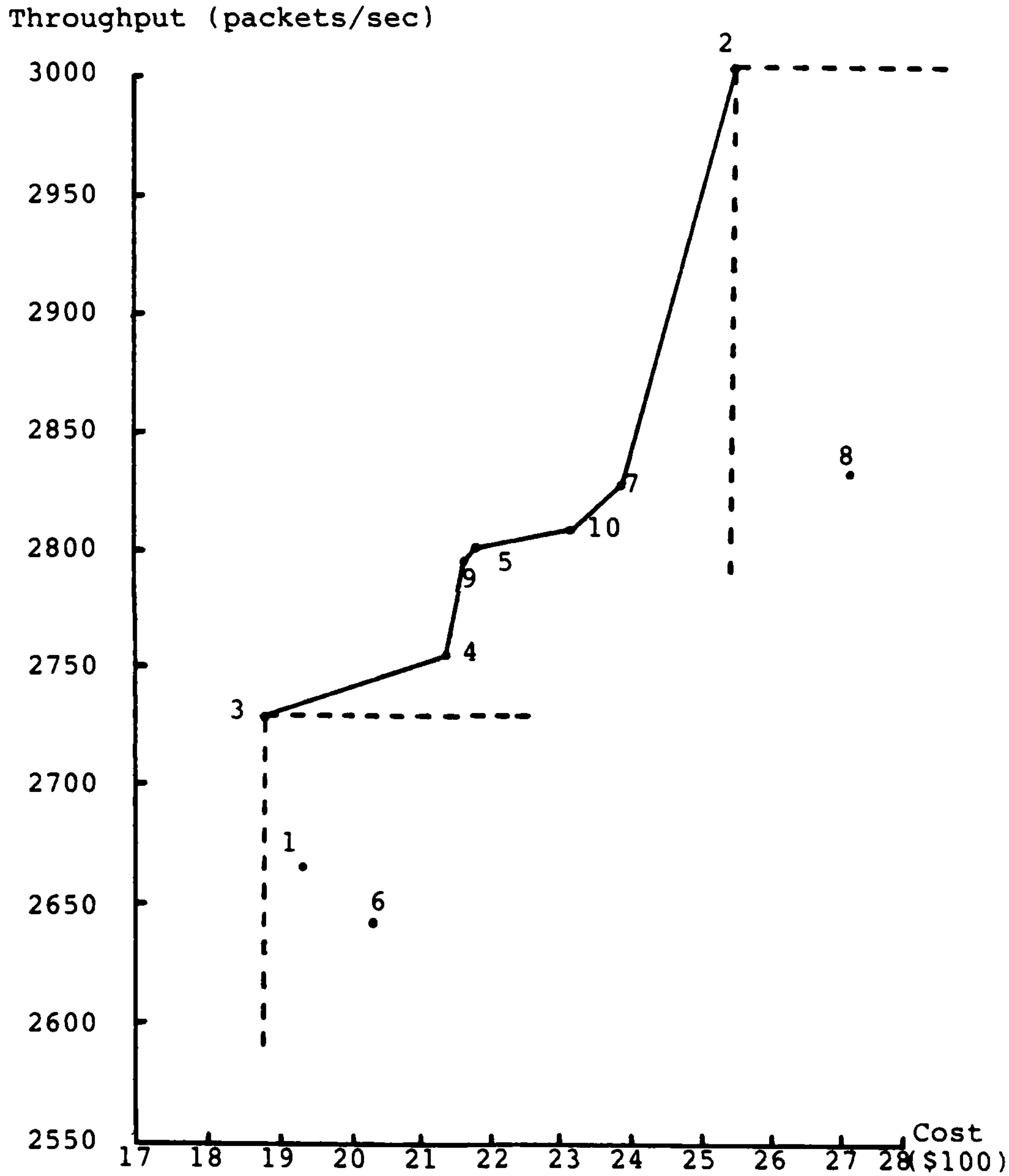


Figure 33. Throughput/cost tradeoff

is reached.

6.2.1 Starting Topology Configuration

The 52 nodes in the network to be optimized consist of 26 circuit switch backbone nodes and 26 peripheral packet switch nodes. As in the 20-node network, the circuit and packet switches are in a one-to-one correspondence, with each packet switch mapping onto a unique circuit switch. The 26 backbone node locations correspond to a 26-node substructure of the ARPANET. These 26 locations are commonly used in the literature [7, 14, 32, 42] when the design and analysis of packet-switched networks is addressed. Figure 34 shows the backbone of the starting topology that was generated by CIRPAC. The integers on the links correspond to line type, and the underlying assumptions (e.g., workload and constraints) are the same as that for the 20-node network discussed in section 6.1. The starting topology contains 27 links and does not satisfy the biconnectivity constraint.

6.2.2 52-Node Network Optimization

The process of optimizing this network with CIRPAC required a total of 45 iterations. Eight iterations were required to reach feasibility, with eight links being added in phase 1. Phase 2 required 37 iterations and two

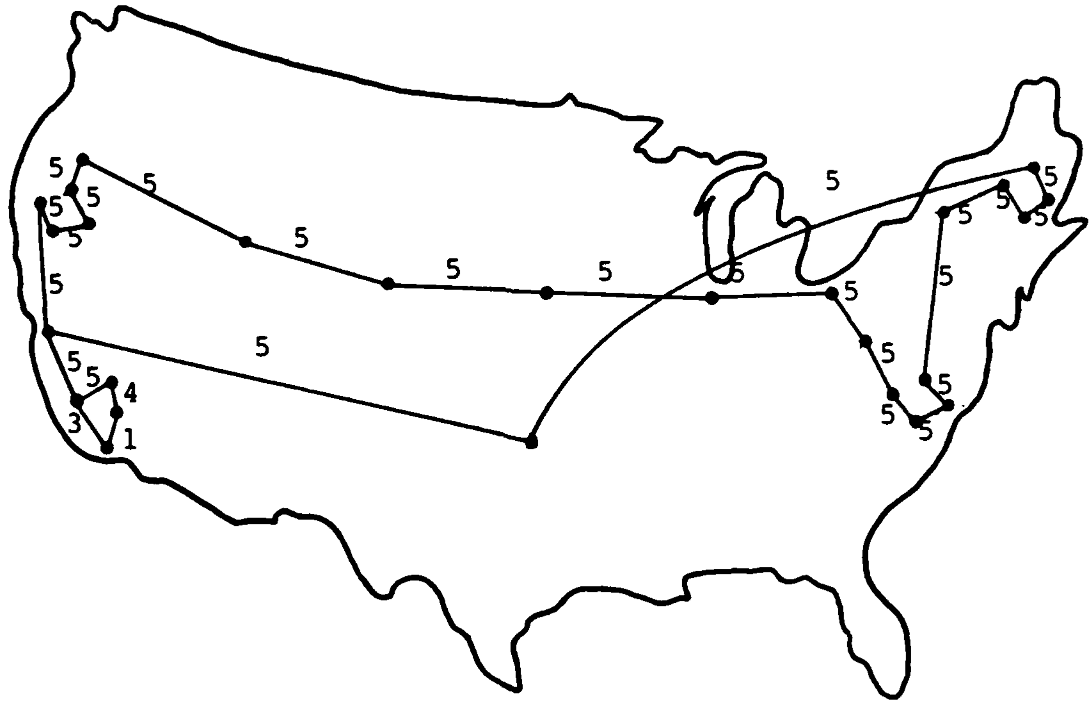


Figure 34. Starting topology for 52-node network

links were deleted in this phase. Figure 35 shows a trace of the cost function from the starting topology to the local minimum obtained at iteration 45. The "hiccups" shown at iterations 22, 24, 26, 28, and 29 represent perturbation failures. That is, the modification to the network produced an infeasible topology, so the procedure reverted back to the "best" topology prior to the next iteration. In all five failures, the constraint failing to be satisfied was the call blocking constraint. The topology representing the local minimum is shown in Figure 36. It has 33 links; 25 of the original links remain (solid lines), and 8 new links have been added (dashed

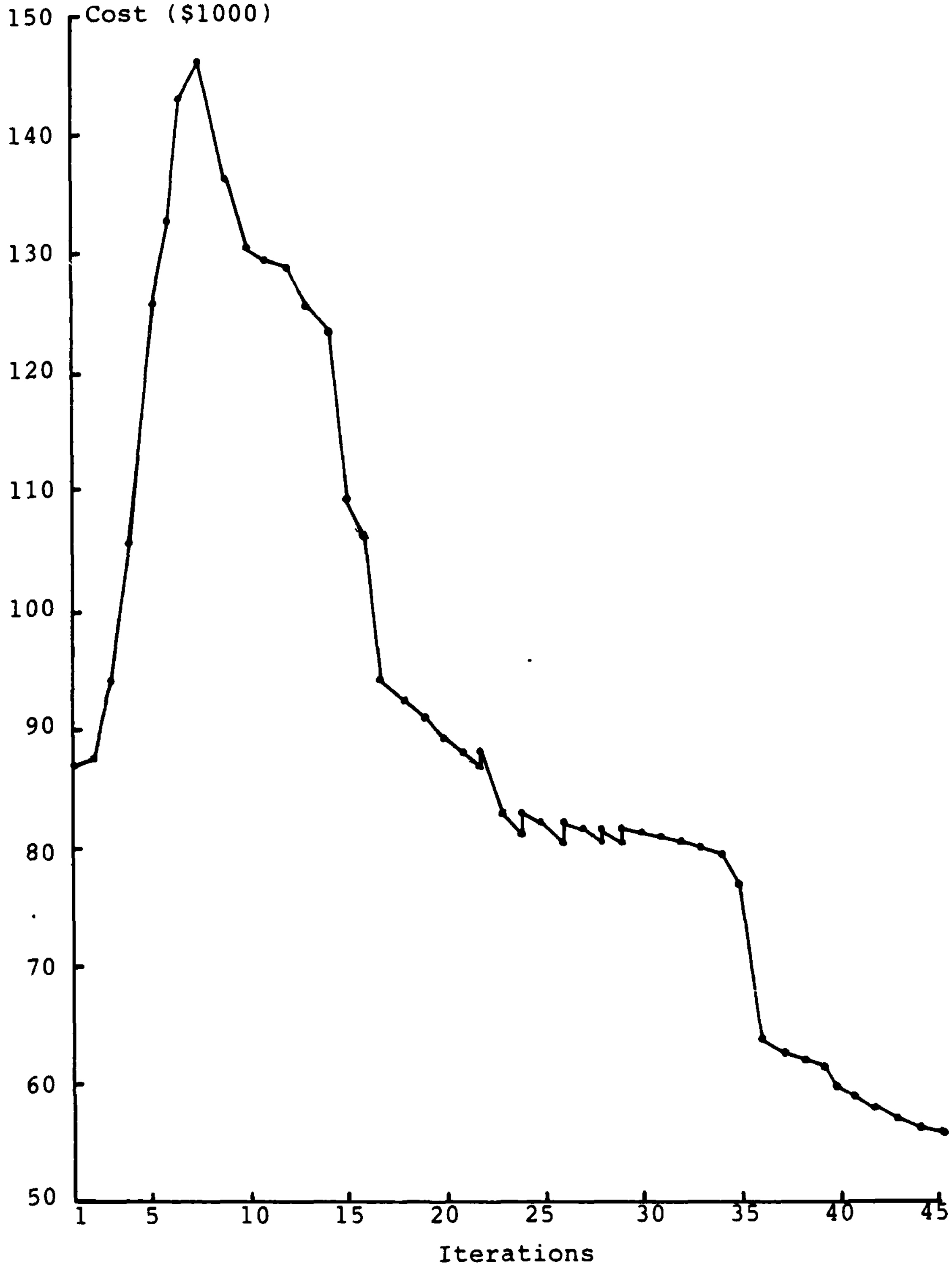


Figure 35. 52-Node optimization: cost trace

lines).

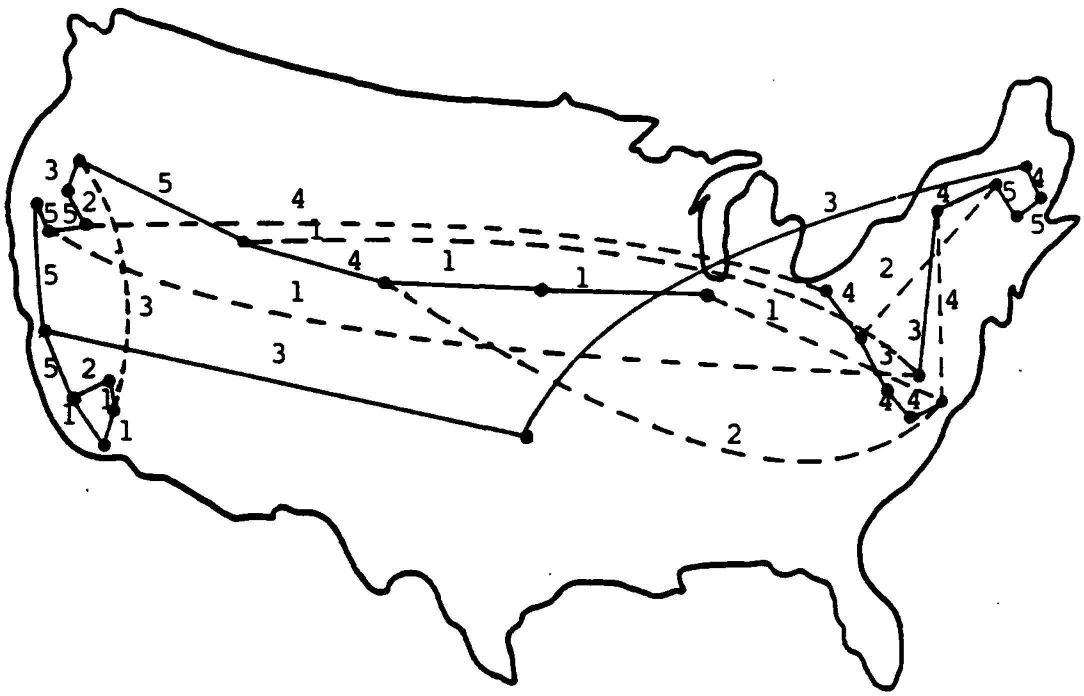


Figure 36. Local minimum topology for 52-node network

6.2.3 Reliability Considerations

In addition to the commonly used biconnectivity constraint, which was originally proposed by Roberts and Wessler [79] and is the criterion implemented in CIRPAC, several other reliability measures are also gaining wider use. Of these, two of the most common measures are the probability of the network being disconnected (PD) and the fraction of nodes unable to communicate (FR) [42]. Both of these measures are obtained by assuming that a link can fail with some nonzero probability PFAIL. If PFAIL is

supplied by the user, CIRPAC will calculate both PD and FR for the local minimum obtained. This is accomplished by performing a Monte Carlo simulation on the suboptimal solution. Each simulation randomly deletes links based on the value of PFAIL, and the fraction of node pairs unable to communicate is calculated for the resulting network. Whether or not the resulting network is disconnected is also recorded. The number of simulations performed is based on the criterion of having a 90% confidence interval such that the true PD is within +/- .02 of the observed PD. For the local minimum topology of the 52-node network in Figure 36 above, PFAIL was assumed to be .05. The results of 1016 simulations on this topology are PD = .0669 and FR = .0097.

6.3 Application to a 26-Node Packet-Switched Network

The absence of empirical data for integrated networks led to the application of CIRPAC to a 26-node ARPA-like network. CIRPAC was neither designed nor developed for packet-switched networks. Hence, any comparisons made between CIRPAC's heuristics and heuristics intended for strictly packet-switched networks have not been validated and should be interpreted with caution.

6.3.1 26-Node ARPANET Design Specifications

The 26 node locations for this design are the

backbone locations shown in Figure 36. These locations are now packet switches, not circuit switches. The design specifications for the 26-node ARPA-like network are as follows [14]:

(1) mean packet delay $\leq$ .2 sec

(2) data packet size = 460 bits

(3) node processing delay = 0 sec for each node

(4) propagation delay = 8.05 μ sec/mile

(5) traffic load ranges from 350 to 750 Kbits/sec and is symmetric and uniformly distributed between node pairs

(6) capacity/cost options:

Line Type	Line Speed (BPS)	Cost($) Per Month/Mile	Fixed Cost($) Per Month
1	9600	.40	1300
2	19200	2.50	1700
3	50000	5.00	1700

The input parameters to CIRPAC have been adjusted to accommodate these design criteria as best possible. Certain approximations have been made. The node processing delay (circuit switching delay) in CIRPAC must be used to approximate the propagation delay of the packet-switched network. This is accomplished by assuming an average distance traveled by each packet (2500 miles) as well as an average number of hops (4) on each routing

path. Also, a mean packet delay of no more than .5 seconds with 1000-bit packets is used [79]. The voice call arrival rate is set to zero since only data packets are traversing this network.

6.3.2 Comparison of Heuristic Algorithms

CIRPAC has been used to generate two local minima. One is for a traffic requirement of 350 Kbits/sec and the other is for a workload of 700 Kbits/sec. Figure 37 illustrates the throughput/cost space for solutions obtained with CIRPAC and other heuristics operating on the 26-node network. The lower bound for optimal solutions is shown as a solid line. The lower bound and all but the two CIRPAC-generated points on the graph are taken from Gerla and Kleinrock [42]. The BXC, CS, and CBE solutions shown on the graph are each dominating solutions. That is, they each represent the lowest cost solution of several local optima obtained at each traffic requirement level shown.

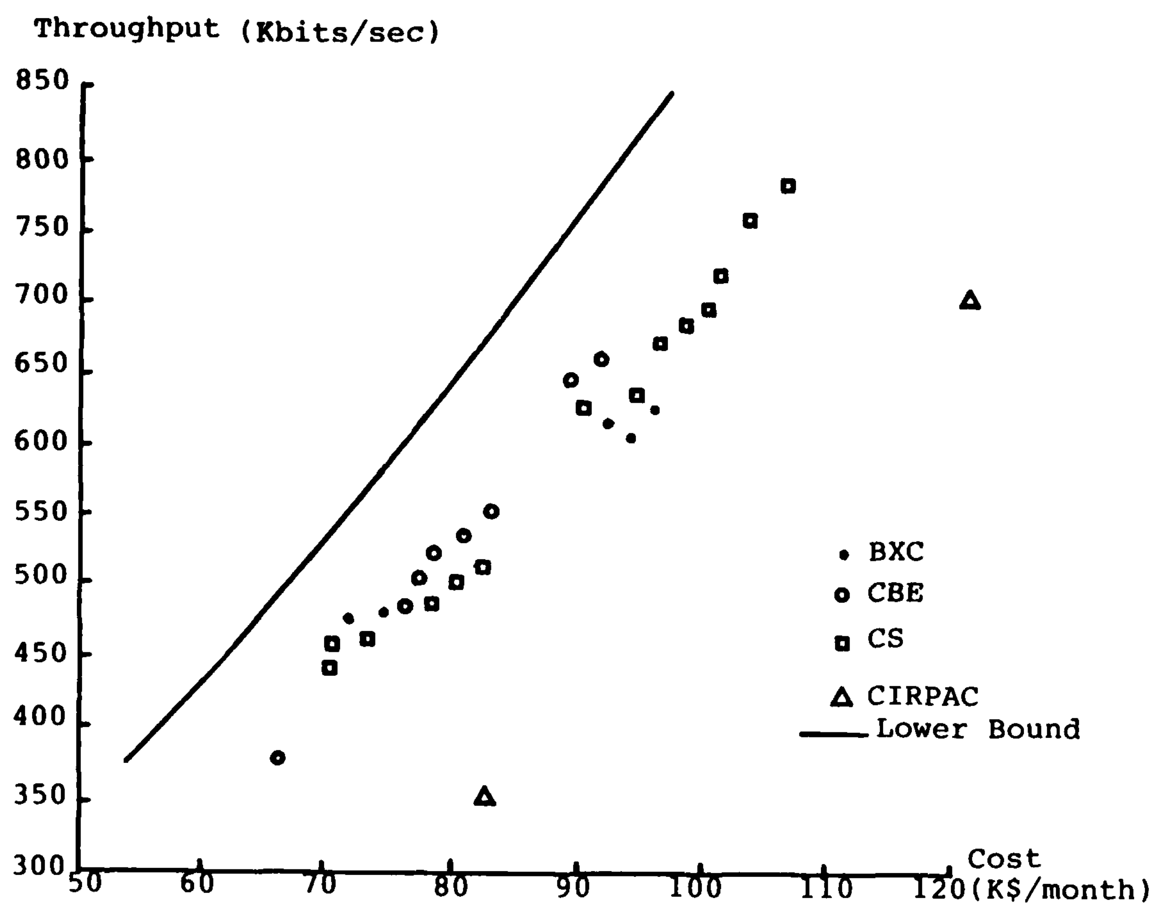


Figure 37. Heuristic solutions for 26-node design

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

7. Conclusions

The primary objective of this research was to design and develop an adaptive topological configuration model which is capable of optimizing the topology of an integrated circuit/packet-switched computer-communication network. Network topology as defined herein was taken to mean the manner in which the nodes of a network are interconnected and how much carrying capacity each connecting link should have.

Initial investigations were directed at identifying what an integrated circuit/packet-switched computer network is. It was shown that such networks are capable of carrying voice and data information simultaneously on their trunk lines. The existence of integrated networks in the near future appears to be a certainty since network designers and managers alike seek to make more efficient use of the information-carrying media available to them. The more recent innovations in the development of very high capacity transmission media, e.g., fiber optics, point to a faster transition to integrated networks than previously anticipated.

Next, an existing model [15] capable of simulating an

integrated network was examined in detail. The slotted envelope (SENET) technique used to superimpose voice and data on the same transmission medium was reviewed, and the voice and data queueing concept implemented in the model was delineated. Modifications to the model were made in order to expand its capability and to make it compatible with the topology manipulation schemes of the network optimization process. Among the major changes were variable link capacity and expanded network performance statistics gathering capabilities.

A detailed network performance analysis was conducted with the simulator to ascertain the relationships between network input parameters and network performance measures. This sensitivity analysis was based on an experimental design which was especially well-suited for computer simulation experimentation. Multiple regression analyses were used to obtain models that describe the relationships between performance and network design parameters. For heavily loaded networks, the performance measures investigated were all seen to be fairly sensitive to network traffic load, link capacity, and network size. Voice call arrival rates tend to dominate the network under the progressive alternate routing strategy implemented in the simulator. An effective range of input parameters was determined for which quadratic response surfaces adequately model network performance.

The exact solution to the topological optimization problem was seen to be intractable for even small networks. This research addresses the topology design problem using an iterative, heuristic approach whereby many suboptimal solutions (local minima) are efficiently generated in lieu of one optimal solution. The iterative scheme integrates the modified simulator as a performance generation device in the middle of a performance feedback loop. The loop consists of three processes that are repeated in turn: topology generation, network performance generation, and performance evaluation. The heuristics, part of the performance evaluation process, determine the direction in which the topology is to be modified. An integrated cut-saturation add heuristic is used to increase total network capacity until a feasible topology is obtained. Then a delete heuristic which preserves biconnectivity is used to reduce cost and increase link utilization. The methodology developed has been designed to be independent of the device which generates network performance data. Hence, if analytical means of providing adequate performance data for an integrated network should become available, only the interface modules would require change.

The methodology has been applied to design problems of varying size. Results indicate that local optima can be obtained in a reasonable number of iterations even with

a relatively small step size. It has been shown how the existence of several local minima can be used to actually increase the flexibility that the decision maker has in making design decisions. The model itself allows for human intervention. Starting topologies may be supplied rather than automatically generated. The number of iterations allowed is easily controlled and the stopping criteria are easily modified. Links and capacities may be forced into the topology if so desired. The methodology represents a viable approach to the design problem, and this research has demonstrated that the model is a flexible tool that can be used to optimize the design of integrated networks.

7.1 Recommendations

More than 95% of the computing costs incurred in this research can be attributed to simulation. Unless dedicated computer facilities are available for large network design, economic considerations prohibit the use of simulation as the only means for generating network performance data. Research needs to be conducted in the area of approximation techniques and analytic procedures that can provide adequate performance data for a given integrated network design specification.

With 1000-node integrated networks in the offing, the need for attacking the network design problem from a

"divide and conquer" aspect is apparent. Decomposition techniques that allow for a hierarchical clustering of nodes need to be investigated. Multiple applications of a design technique on many smaller topologies could realize considerable cost savings over a single application on a large topology. A closely related area requiring attention is the design of gateways between nodal clusters that satisfy rigorous reliability constraints.

Future integrated networks will undoubtedly require priority or preemption schemes for traffic management. The distinction between data classes having different performance requirements also will most certainly be a reality. To be effective, design methodologies must be capable of addressing such issues.

The current approach of using dedicated trunk lines for voice activity needs to be reconsidered. Perhaps the high redundancy of speech can be used to develop speech buffering techniques in which minor delays may be incurred without reducing the quality of transmission. In short, voice digitization algorithms deserve increased attention.

Further investigation into the development of more effective heuristics for integrated network design is needed. The incorporation of more stringent and varied reliability criteria into the heuristic should be considered. This research demonstrated the applicability and effectiveness of specific add and delete heuristics.

Possibly other heuristics which improve the rates of convergence can be developed.

REFERENCES

1. Abramson, N., and Kuo, F. F., (Eds.). Computer-Communication Networks . Prentice Hall, Inc., Englewood Cliffs, NJ, 1973.

2. Aho, A. V., Hopcroft, J. E., and Ullman, J. D. The Design and Analysis of Computer Algorithms . Addison-Wesley Publishing Co., Reading, MA, 1974.

3. Ahuja, V. Design and Analysis of Computer Communication Networks . McGraw-Hill, Inc., New York, NY, 1982.

4. Bell, G. C. More power by networking. IEEE Spectrum 11 , 2 (Feb. 1974), 40-45.

5. Bially, T., and McLaughlin, A. J. Voice communications in integrated digital voice and data networks. IEEE Trans. on Comm. Com-28 , 9 (Sep. 1980), 1478-1488.

6. Boehm, B. W., and Mobley, R. L. Adaptive routing techniques for distributed communications systems. IEEE Trans. on Comm. Techn. Com-17 , 3 (June 1969), 340-349.

7. Boorstyn, R. R., and Frank H. Large-scale network topological optimization. IEEE Trans. on Comm. Com-25 , 1 (Jan. 1977), 29-47.

8. Branscomb, L. M. Trends and developments in computer/ telecommunications technologies. Proc. of the Organization for Economic Cooperation and Development Conference , Paris, France (Feb. 4-6, 1975), 57-77.

9. Cantor, D. G., and Gerla, M. Optimal routing in a packet switched computer network. IEEE Trans. on Comput. C-23 , 10 (Oct. 1974), 1062-1069.

10. Cerf, V. G., and Kahn, R. E. A protocol for packet network interconnection. IEEE Trans. on Comm. Com-22 , 5 (May 1974), 637-648.

11. Chou, W. ACK/TOPS - an integrated network design tool. IEEE 1981 Int. Conf. on Comm. (ICC-81) , Denver, CO (June 14-18, 1981), 4.1.1-4.1.7.

12. Chou, W. (Ed.). Computer Communications, Volume I: Principles . Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

13. Chou, W. Statements made by Dr. Chou at North Carolina State University to the author via telecon (Feb. 17, 1983).

14. Chou, W., and Sapir, D. A generalized cut-saturation algorithm for distributed computer communications network optimization. IEEE 1982 Int. Conf. on Comm. (ICC-82), Philadelphia, PA (June 13-17, 1982), 4C.2.1-4C.2.6.

15. Clabaugh, C. A. Analysis of flow behavior within an integrated computer-communication network. Ph.D. dissertation, Texas A&M University (May 1979).

16. Coviello, G. J., and Lyons, R. E. Conceptual approaches to switching in future military networks. IEEE Trans. on Comm. Com-28, 9 (Sep. 1980), 1491-1498.

17. Coviello, G. J., and Rosner, R. D. Cost considerations for a large data network. 1974 Int. Conf. on Computer Comm. (ICCC-74), Stockholm, Sweden (Aug. 12-14, 1974), 715-720.

18. Coviello, G. J., and Vena, D. A. Integration of circuit/packet switching by a SENET (Slotted Envelope Network) concept. 1975 Nat. Telecomm. Conf. (NTC-75), New Orleans, LA (Dec. 1-3, 1975), 42-12 - 42-17.

19. Cox, J. E. Western union digital services. Proc. of IEEE 60, 11 (Nov. 1972), 1350-1357.

20. Cravis, H. Communications Network Analysis. D. C. Heath and Co., Lexington, MA, 1981.

21. Dirilten, H., and Donaldson, R. W. Topological design of distributed data communication networks using linear regression clustering. IEEE Trans. on Comm. Com-25, 10 (Oct. 1977), 1083-1090.

22. Doll, D. R. Telecommunications turbulence and the computer network evolution. Computer 7, 2 (Feb. 1974), 13-22.

23. Draper, N. R. and Smith, H. Applied Regression Analysis (2nd ed.). John Wiley & Sons, Inc., New York, NY, 1981.

24. Dreyfus, S. E. An appraisal of some shortest-path algorithms. Operations Research 17, 3 (May-June 1969), 395-412.

25. Dysart, H., Krone, M., and Fielding, J. Integrated voice/data private network planning. IEEE 1981 Int. Conf. on Comm. (ICC-81) , Denver, CO (June 14-18, 1981), 4.2.1-4.2.5.

26. Elovitz, H. S., and Heitmeyer, C. L. What is a computer network? 1974 Nat. Telecomm. Conf. (NTC-74) , San Diego, CA (Dec. 2-4, 1974), 1007-1014.

27. Esterling, R., and Hahn, P. A comparison of digital data network switching alternatives. 1975 Nat. Telecomm. Conf. (NTC-75) , New Orleans, LA (Dec. 1-3, 1975), 42-8 - 42-11.

28. Farber, D. J. Networks: an introduction. Datamation 18 , 4 (April 1972), 36-39.

29. Forgie, J. W. Voice conferencing in packet networks. IEEE 1980 Int. Conf. on Comm. (ICC-80) , Seattle, WA (June 12-18, 1980), 21.3.1-21.3.4.

30. Frank, H. Plan today for tommorow's data/voice nets. Data Communications 7 , 9 (Sep. 1978), 51-62.

31. Frank, H., and Chou, W. Network properties of the ARPA computer network. Networks 4 (1974), 213-239.

32. Frank, H., and Chou, W. Topological optimization of computer networks. Proc. of IEEE 60 , 11 (Nov. 1972), 1385-1397.

33. Frank, H., and Gitman, I. Economic analysis of integrated voice and data networks: a case study. Proc. of IEEE 66 , 11 (Nov. 1978), 1549-1570.

34. Freund, R. J., and Minton, P. D. Regression Methods . Marcel Dekker, Inc., New York, NY, 1979.

35. Fuchs, E., and Jackson, P. E. Estimates of distributions of random variables for certain computer communication traffic. Comm. ACM 13 , 12 (Dec. 1970), 752-757.

36. Fultz, G. L., and Kleinrock, L. Adaptive routing techniques for store-and-forward computer communication networks. IEEE 1971 Int. Conf. on Comm. (ICC-71) , Montreal, Canada (June 14-16, 1971), 39-1 - 39-8.

37. Gaines, E. C. Specialized common carriers - competition and alternative. Telecommunications 7 , 9 (Sep. 1973), 15-24.

38. Gallager, R. Distributed network optimization algorithms. IEEE 1979 Int. Conf. on Comm. (ICC-79), Boston, MA (June 10-14, 1979), 43.2.1-43.2.2.

39. Garey, M. R., and Johnson, D. S. Computers and Intractability, A Guide to the Theory of NP-Completeness. W. H. Freeman and Co., New York, NY, 1979.

40. Gerla, M. The design of store-and-forward networks for computer communications. Ph.D. dissertation, School of Eng. and Appl. Sci., Univ. of California, Los Angeles (Jan. 1973).

41. Gerla, M., and Chou, W. Flow control strategies in packet switched computer networks. 1974 Nat. Telecomm. Conf. (NTC-74), San Diego, CA (Dec. 2-4, 1974), 1032-1037.

42. Gerla, M., and Kleinrock, L. On the topological design of distributed computer networks. IEEE Trans. on Comm. Com-25, 1 (Jan. 1977), 48-60.

43. Gerla, M., and Mason, D. Distributed routing in hybrid packet and circuit data networks. IEEE Proc. of Comp. Comm. Networks: COMPCON 78, Washington, DC (Sep. 5-8, 1978), 125-131.

44. Gerla, M., Frank, H., Chou, W., and Eckl, J. A cut saturation algorithm for topological design of packet switched communication networks. 1974 Nat. Telecomm. Conf. (NTC-74), San Diego, CA (Dec. 2-4, 1974), 1074-1079.

45. Gitman, I., Hsieh, W., and Occhiogrosso, B. J. Analysis and design of hybrid switching networks. IEEE Trans. on Comm. Com-29, 9 (Sep. 1981), 1290-1300.

46. Gitman, I., Occhiogrosso, B. J., Hsieh, W., and Frank, H. Sensitivity of integrated voice and data networks to traffic and design variables. Proc. 6th Data Comm. Symposium, Pacific Grove, CA (Nov. 1979), 181-192.

47. Graybeal, W. T., and Pooch, U. W. Simulation: Principles and Methods. Winthrop Publishers, Inc., Cambridge, MA, 1980.

48. Greene, W. H. Optimal routing within large scale distributed computer-communications networks. Ph.D. dissertation, Texas A&M University (May 1978).

49. Gross, D., and Harris, C. M. Fundamentals of Queueing Theory . John Wiley & Sons, Inc., New York, NY, 1974.

50. Gruber, J. G. Delay related issues in integrated voice and data networks. IEEE Trans. on Comm. Com-29 , 6 (June 1981), 786-800.

51. Hoard, B. Integrating voice and data - sharing the lines. Computerworld 15 , 52 (Dec. 28, 1981), 33-35.

52. Hsieh, W., Gitman, I., and Abernathy, J. Topological design issues in network interconnection. IEEE 1977 Int. Conf. on Comm. (ICC-77) , Chicago, IL (June 12-15, 1977), 22.5.126-22.5.131.

53. Hsieh, W., Gitman, I., and Occhiogrosso, B. J. Design of hybrid-switched networks for voice and data. IEEE 1978 Int. Conf. on Comm. (ICC-78) , Toronto, Canada (June 4-7, 1978), 20.1.1-20.1.9.

54. James, R. T., and Muench, P. E. AT&T facilities and services. Proc of IEEE 60 , 11 (Nov. 1972), 1342-1349.

55. Jenny, C. J., Kummerle, K., and Burge, H. Network nodes with integrated circuit/packet switching capabilities. IBM Research Report RZ-720 (Aug. 1975).

56. Kermani, P., and Kleinrock, L. A tradeoff study of switching systems in computer communication networks. IEEE Trans. on Computers C-29 , 12 (Dec. 1980), 1052-1060.

57. Kleinrock, L. Analytic and simulation methods in computer network design. 1970 Spring Joint Computer Conf., AFIPS Conf. Proc. 36 , Atlantic City, NJ (May 5-7, 1970), 569-579.

58. Kleinrock, L., and Kamoun, F. Optimal clustering structures for hierarchical topological design of large computer networks. Networks 10 , 3 (Fall 1980), 221-248.

59. Kozicki, Z., and McGregor, P. V. An approach to computer-aided network design. IEEE 1981 Int. Conf. on Comm. (ICC-81) , Denver, CO (June 14-18, 1981), 4.4.1-4.4.7.

60. Martin, J. Future Developments in Telecommunications (2nd ed.) . Prentice-Hall, Inc., Englewood Cliffs, NJ, 1977.

61. Mathison, S. L., and Walker, P. M. Regulatory and economic issues in computer communications. Proc of IEEE 60, 11 (Nov. 1972), 1254-1272.

62. McAuliffe, D. J. An integrated approach to communications switching. IEEE 1978 Int. Conf. on Comm. (ICC-78), Toronto, Canada (June 4-7, 1978), 20.4.1-20.4.5.

63. McManamon, P. Digital data telecommunications performance criteria. Report AD-A000299, NTIS (June 1974).

64. McQuillan, J. M. Adaptive routing algorithms for distributed computer networks. Report AD-781467, NTIS (May 1974).

65. Metcalfe, R. M. Packet communications. Report AD-771-430, NTIS (Dec. 1973).

66. Montgomery, D. C., and Peck, E. A. Introduction to Linear Regression Analysis. John Wiley & Sons, Inc., New York, NY, 1982.

67. Myers, R. H. Response Surface Methodology. Allyn and Bacon, Inc., Boston, MA, 1971.

68. Naylor, T. H. (Ed.). The Design of Computer Simulation Experiments. Duke University Press, Durham, NC, 1969.

69. Occhiogrosso, B. J., Gitman, I., Hsieh, W., and Frank, H. Performance analysis of integrated switching communications systems. 1977 Nat. Telecomm. Conf. (NTC-77), Los Angeles, CA (Dec. 5-7, 1977), 12:4-1 - 12:4-13.

70. Ostle, B., and Mensing, R. W. Statistics in Research (3rd ed.). The Iowa State University Press, Ames, IO, 1975.

71. Ozarow, L., and DeRosa, J. A combined packet and circuit-switched processing satellite system. IEEE 1979 Int. Conf. on Comm. (ICC-79), Boston, MA (June 10-14, 1979), 24.5.1-24.5.5.

72. Phillips, D. T., and Garcia, A. Fundamentals of Network Analysis. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

73. Pooch, U. W., and Kiemele, M. J. A simulation model for evaluating integrated circuit/packet-switched networks. Presented at Winter Simulation Conf., Arlington, VA (Dec. 12-14, 1983).

74. Pooch, U. W., Greene, W. H., and Moss, G. G. Telecommunications and Networking . Little, Brown and Company, Boston, MA, 1983.

75. Reiser, M. Performance evaluation of data communication systems. IBM Research Report RZ-1092 (Aug. 1981).

76. Rich, M. A., and Schwartz, M. Buffer sharing in computer-communication network nodes. IEEE Trans. on Comm. Com-25 , 9 (Sep. 1977), 958-970.

77. Roberts, L. G. Data by the packet. IEEE Spectrum 11 , 2 (Feb. 1974), 46-51.

78. Roberts, L. G. The evolution of packet switching. Proc. of IEEE 66 , 11 (Nov. 1978), 1307-1313.

79. Roberts, L. G., and Wessler, B. D. Computer network development to achieve resource sharing. 1970 Spring Joint Computer Conf., AFIPS Conf. Proc. 36 , Atlantic City, NJ (May 5-7, 1970), 543-549.

80. Rosner, R. D. A digital data network concept for the Defense Communications Agency. 1973 Nat. Telecomm. Conf. (NTC-73) , Atlanta, GA (Nov. 26-28, 1973), 22C-1 - 22C-6.

81. Rosner, R. D. Large scale network design considerations. 1974 Int. Conf. on Computer Comm. (ICCC-74) , Stockholm, Sweden (Aug. 12-14, 1974), 189-197.

82. Ross, M. J. System engineering of integrated voice and data switches. IEEE 1978 Int. Conf. on Comm. (ICC-78) , Toronto, Canada (June 4-7, 1978), 20.5.1-20.5.4.

83. Ross, M. J., Tabbott, A. C., and Waite, J. A. Design approaches and performance criteria for integrated voice/data switching. Proc of IEEE 65 , 9 (Sep. 1977), 1283-1295.

84. Rudin, H. Studies on the integration of circuit and packet switching. IEEE 1978 Int. Conf. on Comm. (ICC-78) , Toronto, Canada (June 4-7, 1978), 20.2.1-20.2.7.

85. SAS Institute Inc. SAS User's Guide: Basics, 1982 Edition . SAS Institute Inc., Cary, NC, 1982.

86. SAS Institute Inc. SAS User's Guide: Statistics, 1982 Edition . SAS Institute Inc., Cary, NC, 1982.

87. SAS Institute Inc. SAS/GRAPH User's Guide, 1981 Edition . SAS Institute Inc., Cary, NC, 1981.

88. Schmitz, H. G., Saxton, T. L., Huang, C. S., and White, J. A. Application of associative processing techniques to an integrated voice/data switched network. Report AD-A040636, NTIS (June 1976).

89. Schneider, G. M. The VANS system. IEEE Proc of Comp. Comm. Networks: COMPCON 78 , Washington, DC (Sep. 5-8, 1978), 166-174.

90. Schneider, K. S. Integrating voice and data on circuit-switched networks. IEEE Trans. on Aerosp. Electron. Syst. AES-15 , 4 (July 1979), 481-493.

91. Shannon, R. E. Systems Simulation: The Art and Science . Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975.

92. Sharma, R. L., de Sousa, P. T., and Ingle, A. D. Network Systems . Van Nostrand Reinhold Company, New York, NY, 1982.

93. Steiglitz, K., Weiner, P., and Kleitman, D. J. The design of minimum-cost survivable networks. IEEE Trans. on Circuit Theory CT-16 , 4 (Nov. 1969), 455-460.

94. Takehiko, Y., and Shimasaki, N. A study of future integrated service digital networks. 1975 Nat. Telecomm. Conf. (NTC-75) , New Orleans, LA (Dec. 1-3, 1975), 7-1 - 7-6.

95. Tanenbaum, A. S. Computer Networks . Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

96. Thurber, K. J. Circuit switching technology: a state-of-the-art survey. IEEE Proc. of Comp. Comm. Networks: COMPCON 78 , Washington, DC (Sep. 5-8, 1978), 116-124.

97. Weinstein, C., McLaughlin, A., and Bially, T. Efficient multiplexing of voice and data in integrated digital networks. IEEE 1980 Int. Conf. on Comm. (ICC-80) , Seattle, WA (June 8-12, 1980), 21.1.1-21.1.7.

98. Whitney, H. Congruent graphs and the connectivity of graphs. Amer. J. Math. 54 (1932), 150-168.

99. Wilkov, R. S. Analysis and design of reliable computer networks. IEEE Trans. on Comm. Com-20 , 3 (June 1972), 660-678.

100. Worley, A. R. The Datran system. Proc. of IEEE 60 , 11 (Nov. 1972), 1357-1368.

101. Yaged, B. Jr. Minimum cost routing for static network models. Networks 1 (1971), 139-172.

APPENDIX A

Appendix A contains a listing of the adaptive topological configuration model (CIRPAC) developed in this research. Since the program is functionally self-documented, a user should have little difficulty in using or modifying it. As currently dimensioned, the program can accept up to 26 node locations (52 total packet and circuit nodes), 80 trunk lines (160 independent channels), and 8000 slots (an average of 50 slots per channel). If any of these parameter limits are exceeded, the model will indicate to the user which limits have been violated.

```
C************************************************************
C*
C*  DRIVER -  PROGRAM CIRPAC
C*
C*  AN ADAPTIVE TOPOLOGICAL CONFIGURATION OF A CIRCUIT/
C*  PACKET-SWITCHED COMPUTER NETWORK.
C*
C************************************************************
C
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
C
      IADD=0
      IPASS=0
      IRSAVE=0
C READ IN INPUT DATA AND PERFORMANCE CONSTRAINTS.
      CALL INITAL
1     IPASS=IPASS+1
      IF(IPASS.GT.3)GO TO 999
      WRITE(6,201)IPASS
201   FORMAT(1H0,'IPASS=',I4,4(4H****))
      CALL TCM(IPASS)
      CALL INFACE(IPASS)
      CALL SIMULA(IPASS)
      CALL OUTFAC
      CALL PERFRM
      GO TO 1
999   STOP
      END
C
C
C***********************************************************
C*
C*     SUBROUTINE INITAL  -  INITIALIZES THE PROBLEM DOMAIN.
C*
C***********************************************************
C
      SUBROUTINE INITAL
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
C
C READ IN SITE(NODE) INFORMATION
      READ(5,1001)NSITES
      READ(5,1002)((X(I),Y(I)),I=1,NSITES)
      WRITE(6,2001)NSITES
      WRITE(6,2002)
      WRITE(6,2003)((I,X(I),I,Y(I)),I=1,NSITES)
C
C READ IN TRAFFIC OR WORK LOAD INFORMATION
      WRITE(6,2004)(I,I=1,NSITES)
      DO 10 I=1,NSITES
      READ(5,1003)(PSWORK(I,J),J=1,NSITES)
      WRITE(6,2005)I,(PSWORK(I,J),J=1,NSITES)
10    CONTINUE
```

```
      WRITE(6,2006)(I,I=1,NSITES)
      DO 20 I=1,NSITES
      READ(5,1003)(CSWORK(I,J),J=1,NSITES)
      WRITE(6,2005)I,(CSWORK(I,J),J=1,NSITES)
20    CONTINUE
      READ(5,1003)(CSSERV(I),I=1,NSITES)
      WRITE(6,2007)(I,I=1,NSITES)
      WRITE(6,2016)(CSSERV(I),I=1,NSITES)
      READ(5,1004)NPACMS,NBITPK
      WRITE(6,2008)NPACMS,NBITPK
C
C READ IN LINE CAPACITY AND COST INFORMATION
      READ(5,1001)NCAPS
      READ(5,1002)(CCOST(I,1),I=1,NCAPS)
      READ(5,1002)(CCOST(I,2),I=1,NCAPS)
      READ(5,1002)(CCOST(I,3),I=1,NCAPS)
      WRITE(6,2009)
      DO 30 I=1,NCAPS
      WRITE(6,2010)I,(CCOST(I,J),J=1,3)
30    CONTINUE
C
C READ IN THE RELIABILITY CONSTRAINTS
      READ(5,1005)KCON,PFAIL,DSCONL,FRACL
      WRITE(6,2011)KCON,PFAIL,DSCONL,FRACL
C
C READ IN THE DELAY CONSTRAINT
      READ(5,1002)DELAYL
      WRITE(6,2012)DELAYL
C
C READ IN THE THROUGHPUT CONSTRAINT
      READ(5,1002)THRUM
      WRITE(6,2013)THRUM
C
C READ IN THE LINK UTILIZATION CONSTRAINT
      READ(5,1002)UTILM
      WRITE(6,2014)UTILM
C
C READ IN THE CS BLOCKING CONSTRAINT
      READ(5,1002)BLOCKL
      WRITE(6,2015)BLOCKL
C
C READ IN THE TOPOLOGY GENERATOR FLAG AND THE SEED FOR RANDOMIZING
C THE NODES.
      READ(5,1004)ISEED,ISKIP
C
C THE TABLE NEEDTB IS UNIQUE TO THE NETWORK PERFORMANCE GENERATOR
C (I.E., THE SIMULATOR), SO THE FOLLOWING READ OF THE SEEDS FOR
C SIMULA IS EXTRANEOUS TO THE PROBLEM DOMAIN.
      NNODES=2*NSITES
      READ(5,1006)((NEEDTB(I,J),J=1,4),I=1,NNODES)
      WRITE(6,3001)
3001  FORMAT(1HO,5X,'SEED TABLES:')
      WRITE(6,3002)((NEEDTB(I,J),J=1,4),I=1,NNODES)
3002  FORMAT(4(1X,I15))
1001  FORMAT(I3)
1002  FORMAT(6F10.0)
1003  FORMAT(12F6.0)
1004  FORMAT(10I5)
1005  FORMAT(I2,8X,3F10.0)
1006  FORMAT(4(I5,1X))
2001  FORMAT(1H1,40X,'INPUT DATA'/6X,'NO. OF SITES:',I5)
2002  FORMAT(1HO,5X,'SITE COORDINATES:')
2003  FORMAT(2(1X,'X(',I3,')=',F8.2,3X,'Y(',I3,')=',F8.2,5X))
2004  FORMAT(1HO,5X,'PACKET SWITCH TRAFFIC (MESSAGE ARRIVALS/SEC)'/
     1 5X,12(I6))
2005  FORMAT(1X,I4,2X,12(F6.2))
2006  FORMAT(1HO,5X,'CIRCUIT SWITCH TRAFFIC (VOICE ARRIVALS/MIN)'/
     1 5X,12(I6))
```

```
2007  FORMAT(1HO,5X,'CIRCUIT SWITCH SERVICE TIMES (SEC)'/5X,12(I6))
2008  FORMAT(1HO,5X,'AVG. NO. OF PACKETS/MESSAGE =',I5/6X,
     1 'NO. OF BITS/PACKET =',I5)
2009  FORMAT(1HO,5X,'LINE CAPACITY AND COST INFORMATION:'/6X,
     1 'LINE   CAPACITY   COST($) PER   FIXED'/6X,
     2 'TYPE     (BPS)    UNIT LENGTH  COST($)')
2010  FORMAT(6X,I3,2X,F10.0,3X,F10.2,F10.2)
2011  FORMAT(1HO,5X,'RELIABILITY CONSTRAINTS:'/6X,'NODE ',
     1 'CONNECTIVITY (MIN. NO. OF NODES THAT MUST BE DELETED TO ',
     2 'CAUSE A DISCONNECT)=',I3/6X,'PROBABILITY OF LINK FAILURE=',
     3 F5.3/6X,'DISCONNECT PROBABILITY BOUND=',F5.3/6X,
     4 'BOUND ON FRACTION OF NODES UNABLE TO COMMUNICATE=',F5.3)
2012  FORMAT(1HO,5X,'DELAY CONSTRAINT:'/6X,'MEAN PACKET DELAY ',
     1 'SHOULD NOT EXCEED',F8.3,'  SECONDS.')
2013  FORMAT(1HO,5X,'THROUGHPUT CONSTRAINT:'/6X,'AVERAGE NUMBER OF',
     1 ' PACKETS FLOWING THROUGH EACH CHANNEL SHOULD BE AT LEAST',
     2 F10.0,'  PACKETS/SEC.')
2014  FORMAT(1HO,5X,'LINK UTILIZATION CONSTRAINT:'/6X,'MINIMUM ',
     1 'AVERAGE LINK UTILIZATION SHOULD BE',F5.2)
2015  FORMAT(1HO,5X,'CS BLOCKING CONSTRAINT:'/6X,'MAXIMUM ',
     1 'ALLOWABLE AVERAGE SYSTEM BLOCKING IS',F5.2)
2016  FORMAT(5X,12F6.0)
      RETURN
      END
C
C**********************************************************
C*
C*   SUBROUTINE TCM -  THE DRIVER FOR THE TOPOLOGY
C*                     CONFIGURATION MODULE.
C*
C**********************************************************
C
      SUBROUTINE TCM(IPASS)
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
C
      IF(IPASS.GT.1)GO TO 200
      IF(ISKIP.NE.0)GO TO 50
      CALL TOPGEN
      WRITE(6,2003)(I,I=1,NSITES)
2003  FORMAT(1HO,5X,'THE DISTANCE MATRIX:'/8X,2(10I6))
      DO 35 I=1,NSITES
      WRITE(6,2004)I,(DIST(I,J),J=1,NSITES)
2004  FORMAT(7X,I3,2(10F6.0))
35    CONTINUE
      GO TO 200
50    CALL DISGEN
C READ IN THE CONNECTIVITY MATRIX, WHICH IN THIS CASE IS GIVEN.
      DO 55 I=1,NSITES
      READ(5,8001)(CONECT(I,J),J=1,NSITES)
55    CONTINUE
      IF(ISKIP.EQ.1)GO TO 200
C OTHERWISE (IF ISKIP=2), READ IN PERFORMANCE STATS AND CALL
C PERFRM DIRECTLY TO RECONFIGURE THE TOPOLOGY.
      READ(5,8001)NLINKS
      READ(5,8002)AVGU,AVGD,AVGB
      READ(5,8002)(UTIL(I),I=1,NLINKS)
      READ(5,8002)(DELAY(I),I=1,NSITES)
      READ(5,8002)(BLOCK(I),I=1,NSITES)
      READ(5,8001)(NODETB(I,1),I=1,NLINKS)
      READ(5,8001)(NODETB(I,2),I=1,NLINKS)
```

```
      WRITE(6,7005)(I,I=1,NSITES)
      DO 56 I=1,NSITES
      WRITE(6,7006)I,(CONECT(I,J),J=1,NSITES)
56    CONTINUE
      CALL COSTOP
      CALL PERFRM
      GO TO 200
8001  FORMAT(30I2)
8002  FORMAT(6F10.0)
C WRITE OUT THE CONECT MATRIX.
200   WRITE(6,7005)(I,I=1,NSITES)
      DO 67 I=1,NSITES
      WRITE(6,7006)I,(CONECT(I,J),J=1,NSITES)
67    CONTINUE
7005  FORMAT(1H0,5X,'THE CONECT MATRIX NOW LOOKS LIKE:'/6X,30I2)
7006  FORMAT(4X,31I2)
C COMPUTE THE COST OF THE NEW TOPOLOGY.
      CALL COSTOP
      RETURN
      END
C
C*****************************************************
C*
C*   SUBROUTINE TOPGEN - THIS SUBROUTINE GENERATES A
C*
C*   STARTING TOPOLOGY IF ONE IS NEEDED.
C*
C*****************************************************
C
      SUBROUTINE TOPGEN
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1      CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2      PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3      D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4      DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5      NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6      BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC,BCON(26,26),BMAP(26)
      LOGICAL A,B
      REAL RN,BX(26),BY(26)
C
      NM1=NSITES-1
      BCOST=999999999.0
      NSHUFS=1
      WRITE(6,8002)ISEED,NSHUFS
8002  FORMAT(1H0,5X,'ISEED=',I7/6X,'NSHUFS=',I5)
8003  FORMAT(6X,'ISHUF=',I5)
C INITIALIZE THE MAPPING MAP.
      DO 10 I=1,NSITES
      MAP(I)=I
10    CONTINUE
      DO 34 ISHUF=1,NSHUFS
      WRITE(6,8003)ISHUF
C
C RANDOMLY ORDER THE NODES.
      DO 30 I=1,NM1
      CALL RANDUM(RN,ISEED)
      J=I+IFIX((NSITES-I+1)*RN)
      IF(J.EQ.I)GO TO 30
      DUMMYX=X(I)
      DUMMYY=Y(I)
      X(I)=X(J)
      Y(I)=Y(J)
      X(J)=DUMMYX
      Y(J)=DUMMYY
      K=MAP(I)
      MAP(I)=MAP(J)
      MAP(J)=K
```

```
30    CONTINUE
      WRITE(6,2002)
2002  FORMAT(1H ,5X,'THE RANDOMIZED NODES:'/9X,'I',6X,'X(I)',
     1 6X,'Y(I)',2X,'MAP(I)')
      DO 32 I=1,NSITES
      WRITE(6,2001)I,X(I),Y(I),MAP(I)
32    CONTINUE
2001  FORMAT(5X,I5,2F10.2,I5)
C
C GENERATE THE DISTANCE MATRIX DIST
      CALL DISGEN
C       WRITE(6,2003)(I,I=1,NSITES)
C2003   FORMAT(1HO,5X,'THE DISTANCE MATRIX:'/8X,10I6)
C       DO 35 I=1,NSITES
C       WRITE(6,2004)I,(DIST(I,J),J=1,NSITES)
C2004   FORMAT(7X,I3,10F6.2)
C35     CONTINUE
C
C USE A LINK-DEFICIT APPROACH TO INSURE THAT EACH NODE HAS AT
C LEAST DEGREE KCON.
      CALL DEFCIT
C
C DETERMINE IF THE GRAPH IS CONNECTED.
C
C FIRST, INITIALIZE THE LOGICAL MATRIX A TO THE CONECT MATRIX.
      DO 40 I=1,NM1
      A(I,I)=.FALSE.
      L=I+1
          DO 39 J=L,NSITES
          IF(CONECT(I,J).EQ.O)GO TO 38
          A(I,J)=.TRUE.
          A(J,I)=.TRUE.
          GO TO 39
38        A(I,J)=.FALSE.
          A(J,I)=.FALSE.
39        CONTINUE
40    CONTINUE
      A(NSITES,NSITES)=.FALSE.
C
C COPY A TO WORK AREA B.
42    DO 43 I=1,NSITES
      DO 43 J=1,NSITES
      B(I,J)=A(I,J)
43    CONTINUE
C       WRITE(6,7001)(I,I=1,NSITES)
C7001   FORMAT(1HO,5X,'A=CONECT:'/6X,30I2)
C7002   FORMAT(4X,I3,10L3)
C       DO 78 I=1,NSITES
C       WRITE(6,7002)I,(A(I,J),J=1,NSITES)
C78     CONTINUE
C
C COMPUTE THE TRANSITIVE CLOSURE OF THE GRAPH(NETWORK).
      DO 50 I=1,NSITES
      DO 50 J=1,NSITES
      IF(.NOT.B(J,I))GO TO 50
          DO 45 K=1,NSITES
          B(J,K)=B(J,K).OR.B(I,K)
45        CONTINUE
50    CONTINUE
C       WRITE(6,7003)(I,I=1,NSITES)
C7003   FORMAT(1HO,5X,'B=TRANS. CLOSURE OF A:'/6X,30I2)
C       DO 79 I=1,NSITES
C       WRITE(6,7002)I,(B(I,J),J=1,NSITES)
C79     CONTINUE
C
C NOW EXAMINE B TO SEE IF ANY NODE PAIRS ARE DISCONNECTED.  IF
C THERE ARE ANY SUCH NODE PAIRS, FIND THE NODE PAIR HAVING THE
C SHORTEST DISTANCE AND CONNECT THEM DIRECTLY WITH AN ARC.
```

```
C RETURN TO 42, CALCULATE THE TRANSITIVE CLOSURE AGAIN, AND
C CHECK AGAIN TO SEE IF THE GRAPH IS NOW CONNECTED.  REPEAT UNTIL
C THE GRAPH IS CONNECTED.
      DD=999999.
      II=0
      JJ=0
      DO 60 I=1,NM1
      L=I+1
          DO 55 J=L,NSITES
          IF(B(I,J))GO TO 55
          IF(DIST(I,J).GE.DD)GO TO 55
          DD=DIST(I,J)
          II=I
          JJ=J
55        CONTINUE
60    CONTINUE
      IF(II.EQ.0)GO TO 500
      CONECT(II,JJ)=1
      CONECT(JJ,II)=1
      A(II,JJ)=.TRUE.
      A(JJ,II)=.TRUE.
C      WRITE(6,7004)II,JJ,DD
C7004  FORMAT(1H0,5X,'LINK',2I3,'  IS ADDED, HAVING DISTANCE',F10.2)
      GO TO 42
500   CONTINUE
C
C WE NOW HAVE A CONNECTED NETWORK (NOT NECESSARILY K-CONNECTED,
C HOWEVER), SO WE NOW CAN ASSIGN STARTING CAPACITIES TO EACH LINK
C IN THE NETWORK BASED ON THE FOLLOWING CRITERIA:
C  (1) FIND THE SHORTEST PATH BETWEEN NODES X AND Y-CALL IT XABY.
C  (2) ADD THE MAX WORKLOAD OF (X TO Y, Y TO X) TO EACH OF THE
C      LINKS XA, AB, AND BY ON THE PATH.
C  (3) REPEAT THIS FOR EACH NODE PAIR, FINALLY ASSIGNING ONE OF
C      THE DISCRETE CAPACITIES AVAILABLE(1,2,...,NCAPS) TO EACH LINK.
      CALL FLOYDS
      CALL CASIGN
      CALL COSTOP
      IF(COST.GE.BCOST)GO TO 34
      BCOST=COST
      IBEST=ISHUF
      DO 65 I=1,NSITES
      BMAP(I)=MAP(I)
      BX(I)=X(I)
      BY(I)=Y(I)
      DO 65 J=1,NSITES
      BCON(I,J)=CONECT(I,J)
65    CONTINUE
34    CONTINUE
      DO 66 I=1,NSITES
      MAP(I)=BMAP(I)
      X(I)=BX(I)
      Y(I)=BY(I)
      DO 66 J=1,NSITES
      CONECT(I,J)=BCON(I,J)
66    CONTINUE
      CALL DISGEN
C WRITE OUT THE BEST STARTING TOPOLOGY CONECT MATRIX.
      WRITE(6,7005)IBEST,(I,I=1,NSITES)
      DO 67 I=1,NSITES
      WRITE(6,7006)I,(CONECT(I,J),J=1,NSITES)
67    CONTINUE
7005  FORMAT(1H0,5X,'THE BEST STARTING TOPOLOGY FOUND IS ',
     1 'CONECT, WHICH WAS OBTAINED AFTER SHUFFLE NO.',
     2 I5/6X,30I2)
7006  FORMAT(4X,31I2)
      WRITE(6,2002)
      DO 68 I=1,NSITES
      WRITE(6,2001)I,X(I),Y(I),MAP(I)
```

```
68    CONTINUE
      RETURN
      END
C
C*************************************************************
C*
C*    SUBROUTINE DISGEN - GENERATES THE DISTANCE MATRIX.
C*
C*************************************************************
C
      SUBROUTINE DISGEN
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1      CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2      PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3      D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4      DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5      NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6      BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
C
      NM1=NSITES-1
      DO 40 I=1,NM1
      DIST(I,I)=0.0
      L=I+1
         DO 35 J=L,NSITES
         DIST(I,J)=SQRT((X(I)-X(J))**2+(Y(I)-Y(J))**2)
         DIST(J,I)=DIST(I,J)
35       CONTINUE
40    CONTINUE
      DIST(NSITES,NSITES)=0.0
      RETURN
      END
C
C************************************************************
C*
C*    SUBROUTINE RANDUM - IS A RANDOM NUMBER GENERATOR
C*                        FOR THE PROBLEM DOMAIN.
C*
C************************************************************
C
      SUBROUTINE RANDUM(RN,ISEED)
      ISEED=125*(ISEED+1)
      ITRUNK=ISEED/8192
      ISEED=ISEED-8192*ITRUNK
      RN=(ISEED+0.5)/8192.
      RETURN
      END
C************************************************************
C*
C*    SUBROUTINE DEFCIT - A LINK DEFICIT APPROACH TO
C*                        ASSIGNING LINKS IN A NETWORK
C*
C************************************************************
C
      SUBROUTINE DEFCIT
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1      CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2      PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3      D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4      DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5      NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6      BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
      INTEGER P,Q
C
C INITIALIZE LINKED LIST, DIST MATRIX, AND CONECT MATRIX.
```

```
      KCON=2
C      WRITE(6,2001)
      DO 5 I=1,NSITES
      IP1=I+1
      IM1=I-1
      DEG(I)=0
      SUCC(I)=IP1
      PRED(I)=IM1
      HEAD(I)=0
C      WRITE(6,2002)I,DEG(I),SUCC(I),PRED(I),HEAD(I)
         DO 4 K=1,NSITES
         CONECT(I,K)=0
4        CONTINUE
5     CONTINUE
      HEAD(1)=1
      SUCC(NSITES)=0
C
C FIND THE FIRST NODE, P, OF THE NEXT LINK(PQ) TO BE ADDED.
      LISTP=1
10    P=HEAD(LISTP)
C
C BEGIN SEARCH FOR Q(TO GET LINK PQ)
C WHEN Q>0, A VIABLE LINK HAS BEEN FOUND.
      Q=0
      LISTQ=LISTP
      DISMIN=999999.
55    I=HEAD(LISTQ)
C IF STILL NOT AT END OF LIST, GO TO 60
56    IF(I.GT.0)GO TO 60
      IF(Q.GT.0)GO TO 125
      LISTQ=LISTQ+1
      GO TO 55
60    IF(I.NE.P)GO TO 80
61    I=SUCC(I)
      GO TO 56
80    IF(CONECT(I,P).NE.0)GO TO 61
100   IF(DIST(I,P).GE.DISMIN)GO TO 61
      Q=I
      DISMIN=DIST(I,P)
      GO TO 61
C
C ADD ARC PQ TO CONECT AND UPDATE LINKED LISTS.
125   CONECT(P,Q)=1
      CONECT(Q,P)=1
      DEG(P)=DEG(P)+1
      DEG(Q)=DEG(Q)+1
      LP=DEG(P)+1
      LQ=DEG(Q)+1
C DELETE P FROM LISTP AND INSERT P INTO LP
      CALL DELETE(P,LISTP)
      CALL INSERT(P,LP)
C DELETE Q FROM LISTQ AND INSERT Q INTO LQ
      CALL DELETE(Q,LISTQ)
      CALL INSERT(Q,LQ)
C
C      WRITE(6,2001)
C      DO 70 I=1,NSITES
C      WRITE(6,2002)I,DEG(I),SUCC(I),PRED(I),HEAD(I),LISTP,
C     1 LISTQ,LP,LQ
C70    CONTINUE
C
C CHECK TO SEE IF LISTP IS EMPTY. IF IT IS, CHECK TO SEE IF IT
C IS THE LAST LIST NEEDED TO BE CHECKED TO INSURE EACH NODE HAS
C DEGREE AT LEAST KCON.
      IF(HEAD(LISTP).NE.0)GO TO 10
      IF(LISTP.EQ.KCON)GO TO 999
      LISTP=LISTP+1
      GO TO 10
```

```
999   CONTINUE
C      WRITE(6,2003)(I,I=1,NSITES)
C      DO 120 I=1,NSITES
C      WRITE(6,2004)I,(CONECT(I,J),J=1,NSITES)
C120   CONTINUE
C
1001  FORMAT(6F10.0)
2001  FORMAT(1H0,5X,'NODE   DEG  SUCC  PRED  HEAD LISTP ',
     1 'LISTQ    LP    LQ')
2002  FORMAT(4X,9I6)
2003  FORMAT(1H0,5X,'CONNECTIVITY MATRIX:'/6X,30I2)
2004  FORMAT(4X,11I3)
      RETURN
      END
C
C************************************************************
C*
C*  SUBROUTINE DELETE - REMOVES AN ITEM FROM A LINKED LIST
C*
C************************************************************
C
      SUBROUTINE DELETE(P,LISTP)
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
      INTEGER P
C
      I=PRED(P)
      J=SUCC(P)
      IF((I.NE.0).AND.(J.NE.0))GO TO 4
      IF((I.EQ.0).AND.(J.EQ.0))GO TO 3
      IF(I.EQ.0)GO TO 2
C
C P IS THE LAST ELEMENT IN THE LIST, BUT IT HAS A PREDECESSOR.
1     SUCC(I)=0
      RETURN
C
C P IS THE FIRST ELEMENT IN THE LIST, AND IT HAS A SUCCESSOR.
2     HEAD(LISTP)=J
      PRED(J)=0
      RETURN
C
C P IS THE ONLY ELEMENT IN THE LIST.
3     HEAD(LISTP)=0
      RETURN
C
C P IS SOMEWHERE IN THE MIDDLE OF THE LIST.
4     SUCC(I)=J
      PRED(J)=I
      RETURN
      END
C
C************************************************************
C*
C*   SUBROUTINE INSERT - INSERTS AN ITEM INTO A LINKED LIST
C*
C************************************************************
C
      SUBROUTINE INSERT(P,LP)
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
```

```
     3        D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4        DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5        NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6        BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
      INTEGER P
C
.     K=HEAD(LP)
      IF(K.NE.0)GO TO 40
C
C LIST IS CURRENTLY EMPTY.
      HEAD(LP)=P
      PRED(P)=0
      SUCC(P)=0
      RETURN
C
C PUT P IN THE PROPER POSITION IN THE LINKED LIST, WHICH IS
C ORDERED BY NODE NUMBER.
40    IF(P.LT.K)GO TO 50
      IF(SUCC(K).EQ.0)GO TO 45
      K=SUCC(K)
      GO TO 40
C
C PUT P AT THE END OF THE LIST.
45    SUCC(P)=0
      PRED(P)=K
      SUCC(K)=P
      RETURN
C
C INSERT P PRIOR TO K AND AFTER L
50    L=PRED(K)
      PRED(K)=P
      SUCC(P)=K
      PRED(P)=L
      IF(L.NE.0)SUCC(L)=P
      IF(L.EQ.0)HEAD(LP)=P
      RETURN
      END
C
C************************************************************
C*
C*   SUBROUTINE COSTOP - CALCULATES THE COST OF THE
C*                       CURRENT TOPOLOGY.
C*
C************************************************************
C
      SUBROUTINE COSTOP
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1        CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2        PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3        D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4        DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5        NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6        BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
C
      COST=0.0
      DO 40 I=1,NSITES
      L=I+1
          DO 30 J=L,NSITES
          IF(CONECT(I,J).EQ.0)GO TO 30
          K=CONECT(I,J)
          COST=COST+DIST(I,J)*CCOST(K,2)+CCOST(K,3)
30        CONTINUE
40    CONTINUE
      WRITE(6,2001)COST
```

```
2001  FORMAT(1HO,5X,'THE COST($) OF THIS TOPOLOGY IS:',F12.2)
      RETURN
      END
C*****************************************************
C*
C*   SUBROUTINE FLOYDS - COMPUTES SHORTEST PATH BETWEEN
C*                       EVERY PAIR OF VERTICES.
C*
C*****************************************************
C
      SUBROUTINE FLOYDS
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
C
C INITIALIZE THE SHORTEST DISTANCE MATRIX, D.
      DO 11 I=1,NM1
      D(I,I)=0.0
      L=I+1
         DO 6 J=L,NSITES
         IF(CONECT(I,J))3,3,4
3        D(I,J)=999999.0
         GO TO 7
4        D(I,J)=DIST(I,J)
7        D(J,I)=D(I,J)
6        CONTINUE
11    CONTINUE
     .D(NSITES,NSITES)=0.0
C
C INITIALIZE THE INTERMEDIATE NODE MATRIX, INODE.
      DO 20 I=1,NSITES
      DO 20 J=1,NSITES
      INODE(I,J)=J
20    CONTINUE
C
C TRIPLE OPERATION TO UPDATE THE D AND INODE MATRICES.
      DO 30 K=1,NSITES
      DO 30 I=1,NSITES
      DO 30 J=1,NSITES
      IF((I.EQ.K).OR.(I.EQ.J).OR.(J.EQ.K))GO TO 30
      XX=D(I,K)+D(K,J)
      YY=D(I,J)
      IF(YY.LE.XX)GO TO 30
      D(I,J)=XX
      INODE(I,J)=INODE(I,K)
30    CONTINUE
C
C PRINT OUT D AND INODE.
C      WRITE(6,2005)(I,I=1,NSITES)
C      DO 40 I=1,NSITES
C      WRITE(6,2006)I,(D(I,J),J=1,NSITES)
C40    CONTINUE
C      WRITE(6,2007)(I,I=1,NSITES)
C      DO 50 I=1,NSITES
C      WRITE(6,2008)I,(INODE(I,J),J=1,NSITES)
C50    CONTINUE
2005  FORMAT(1HO,5X,'SHORTEST DISTANCE MATRIX D:'/8X,10I8)
2006  FORMAT(7X,I3,10F8.2)
2007  FORMAT(1HO,5X,'INTERMEDIATE NODE MATRIX INODE:'/7X,10I5)
2008  FORMAT(2X,11I5)
      RETURN
      END
```

```
C
C****************************************************************
C*
C*  SUBROUTINE CASIGN - ASSIGNS CAPACITIES TO THE LINKS IN
C*                      CONECT
C*
C****************************************************************
C
      SUBROUTINE CASIGN
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
C
C INITIALIZE MATRIX D TO ALL O'S.
      DO 10 I=1,NSITES
      DO 10 J=1,NSITES
      D(I,J)=0.0
10    CONTINUE
C
C FOR EACH NODE PAIR, X AND Y, TRACE ITS SHORTEST PATH, PLACING
C THE MAXIMUM WORKLOAD OF XY VS. YX ON EACH LINK OF THE PATH
C FROM X TO Y.
      DO 100 I=1,NM1
      II=MAP(I)
      L=I+1
          DO 80 J=L,NSITES
          JJ=MAP(J)
          PSLOAD=AMAX1(PSWORK(II,JJ),PSWORK(JJ,II))*NPACMS*NBITPK
          CSLOAD=AMAX1(CSWORK(II,JJ),CSWORK(JJ,II))*AMAX1(CSSERV(II),
     1    CSSERV(JJ))*32000./60.
C
C NOW ADD THE WORKLOAD (BPS) TO EACH LINK IN THE PATH
C FROM I TO J.  THE FACTOR 1.2 IS FOR ACKNOWLEDGMENTS IN
C THE DATA CASE, AND 2.0 IS FOR FULL VOICE CIRCUIT COMPLETION
C SINCE THE ARRIVAL RATE MEANS INITIATING CALLS ONLY.
          K=I
20        M=INODE(K,J)
          D(K,M)=D(K,M)+1.2*PSLOAD+2.0*CSLOAD
          D(M,K)=D(K,M)
          IF(M.EQ.J)GO TO 80
          K=M
          GO TO 20
80        CONTINUE
100   CONTINUE
C
C PRINT OUT THE ESTIMATED LOADING (IN BPS) OF THE CONECT MATRIX.
C       WRITE(6,2001)(I,I=1,NSITES)
C       DO 120 I=1,NSITES
C       WRITE(6,2002)I,(D(I,J),J=1,NSITES)
C120    CONTINUE
C2001   FORMAT(1HO,5X,'ESTIMATED LOADING OF CONECT (IN BPS)=D:'/5X,
C       1 2(5I10))
C2002   FORMAT(2X,I3,2(5F10.0))
C
C NOW, BASED ON THE LOAD FACTOR, SELECT FOR EACH LINK THE INTEGER
C CAPACITY JUST LARGE ENOUGH TO CARRY THE ESTIMATED LOAD FOR THAT
C LINK.
      DO 200 I=1,NM1
      L=I+1
          DO 150 J=L,NSITES
          IF(CONECT(I,J).EQ.0)GO TO 150
              DO 140 K=1,NCAPS
```

```
            IF(D(I,J).GT.CCOST(K,1))GO TO 140
            CONECT(I,J)=K
            CONECT(J,I)=K
            GO TO 150
140         CONTINUE
        CONECT(I,J)=NCAPS
        CONECT(J,I)=NCAPS
150     CONTINUE
200   CONTINUE
      RETURN
      END
C*************************************************************
C*
C*  SUBROUTINE INFACE ACCOMPLISHES THE TRANSFORMATION FROM
C*  THE PROBLEM DOMAIN TO THE SIMULATION DOMAIN, SO IT ACTS
C*  AS AN INTERFACE BETWEEN THE TOPOLOGY GENERATOR AND THE
C*  NETWORK TOPOLOGY PERFORMANCE GENERATOR WHICH, IN THIS
C*  APPLICATION, IS A SIMULATOR.
C*
C*************************************************************
C
      SUBROUTINE INFACE(IPASS)
      IMPLICIT INTEGER (A-S)
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      LOGICAL A,B
      REAL PSWORK,CSWORK,CSSERV,CCOST,DIST,D,PFAIL,DSCONL,
     1     FRACL,DELAYL,BLOCKL
      REAL DELAY,BLOCK,AVGT,AVGU,AVGTPS,AVGD,AVGB
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      DIMENSION CAVAIL(26),DIMLIM(3),NSLOTS(5)
      DATA DIMLIM/52,160,8000/,NSLOTS/24,30,36,48,60/
C
C THIS SECTION INITIALIZES THE SEED TABLES AND PARAM(1).
      PARAM(1)=2*NSITES
      ALLDST=PARAM(1)
      DO 25 I=1,ALLDST
          DO 20 J=1,4
          SEEDTB(I,J)=NEEDTB(I,J)
20        CONTINUE
25    CONTINUE
C
C IF WE ARE RETURNING FROM A DELETION FAILURE, WE MUST RECONSTRUCT
C THE ROUTING TABLES NO MATTER WHAT VALUE IADD HAS.
      IF(IRSAVE.EQ.2)GO TO 4
      IRSAVE=0
      IF((IPASS.GT.1).AND.(IADD.EQ.0))GO TO 17
C
C THIS SECTION DETERMINES THE PARAMETER VALUES NEEDED BY SIMULA.
C
C TO DETERMINE PARAM(2), WE CONVERT THE CONECT MATRIX TO THE
C BOOLEAN MATRIX A AND DETERMINE HOW MANY LINKS ARE IN THE GRAPH.
4     IRSAVE=0
      NLINKS=0
      DO 10 I=1,NM1
      A(I,I)=.FALSE.
```

```
      L=I+1
          DO 5 J=L,NSITES
          A(I,J)=.FALSE.
          A(J,I)=.FALSE.
          IF(CONECT(I,J).EQ.0)GO TO 5
          A(I,J)=.TRUE.
          A(J,I)=.TRUE.
          NLINKS=NLINKS+1
5         CONTINUE
10    CONTINUE
      A(NSITES,NSITES)=.FALSE.
      PARAM(2)=2*NLINKS
C
C PARAM(3) WILL BE EXPANDED LATER TO CORRESPOND TO TOTAL
C NETWORK CAPACITY IN SLOTS.
      PARAM(4)=3
      PARAM(5)=10
      PARAM(6)=50
C
C FOR NOW WE ASSUME A UNIFORMLY DISTRIBUTED NETWORK IN THE
C CALCULATION OF CS AND PS.
      X7=0.0
      X8=0.0
      DO 15 I=1,NSITES
      X7=X7+CSWORK(1,I)
      X8=X8+PSWORK(1,I)
15    CONTINUE
      PARAM(7)=X7
      PARAM(8)=X8
      PARAM(9)=0
      PARAM(10)=420000
      PARAM(11)=99000
      PARAM(12)=32000
      PARAM(13)=1800
      PARAM(14)=CSSERV(1)
      PARAM(15)=NBITPK
      PARAM(16)=10
C
C THIS SECTION SETS UP THE ROUTING TABLES. IF A LINK DELETION IS TO
C OCCUR, NOTE THIS FACT BY SETTING IRSAVE TO 1.
      IF(IADD.EQ.-1)IRSAVE=1
      DO 30 I=1,NSITES
      DESTAB(I,I)=0
      DSTALT(I,I)=0
      CSNODE=I+NSITES
          DO 29 J=1,NSITES
          K=NSITES+J
          IF(I.EQ.J)GO TO 28
          DESTAB(I,J)=CSNODE
          DSTALT(I,J)=CSNODE
28        DESTAB(I,K)=0
          DSTALT(I,K)=0
29        CONTINUE
30    CONTINUE
C
C ESTABLISH THE CHANNEL NUMBERS (N) BETWEEN NODES IN LINKTB AND
C ALSO THE IMMEDIATE DESTINATION NODE TABLE NODCHL, AS WELL AS
C ESTABLISHING VARIABLE PARAM(3) OR LINK CAPACITIES FOR EACH LINK.
C THESE VARIABLE LINK CAPACITIES ARE DENOTED BY THE PARM3 ARRAY.
      N=1
      DO 40 I=1,NM1
      LINKTB(I,I)=0
      L=I+1
          DO 35 J=L,NSITES
          LINKTB(I,J)=0
          LINKTB(J,I)=0
          IF(CONECT(I,J).EQ.0)GO TO 35
          LINKTB(I,J)=N
```

```
            NODCHL(N)=J+NSITES
            SORCHL(N)=I+NSITES
            LINKTB(J,I)=N+1
            NODCHL(N+1)=I+NSITES
            SORCHL(N+1)=J+NSITES
            N=N+2
35          CONTINUE
40    CONTINUE
      LINKTB(NSITES,NSITES)=0
C
C LINKTB IS USED HERE AS A WORKING TABLE TO COMPLETE THE DESTAB
C AND DSTALT TABLES.  NOW WE CALL FLOYDS TO GENERATE INODE, THE
C INTERMEDIATE NODE MATRIX FOR SHORTEST PATHS.  INODE PROVIDES AN
C INDEX INTO LINKTB WHICH THEN GIVES THE PROPER CHANNEL TO TAKE
C OUT OF A NODE.  THIS WILL PROVIDE THE ENTRY INTO DESTAB.  THE
C ALTERNATE PATH (DSTALT) FROM A NODE WILL BE CHOSEN AS THE PATH
C IN LINKTB THAT WILL MINIMIZE THE NUMBER OF HOPS TO THE
C DESTINATION NODE.
      CALL DISGEN
      CALL FLOYDS
C
      DO 60 I=1,NSITES
      II=I+NSITES
            DO 59 J=1,NSITES
            JJ=J+NSITES
            PTR=INODE(I,J)
            CHAN=LINKTB(I,PTR)
            DESTAB(II,J)=CHAN
            DESTAB(II,JJ)=CHAN
            IF(CHAN.EQ.0)GO TO 58
            NAVAIL=0
                  DO 55 K=1,NSITES
                  IF(LINKTB(I,K).EQ.0)GO TO 55
                  IF(LINKTB(I,K).EQ.CHAN)GO TO 55
                  NAVAIL=NAVAIL+1
                  CAVAIL(NAVAIL)=LINKTB(I,K)
55                CONTINUE
            IF(NAVAIL.EQ.0)GO TO 58
            IF(NAVAIL.GT.1)GO TO 56
            DSTALT(II,J)=CAVAIL(1)
            DSTALT(II,JJ)=CAVAIL(1)
            GO TO 59
C
C WE NOW HAVE A CHOICE SO WE DETERMINE THE LINK BASED ON A
C MINIMUM NUMBER OF HOPS.
56          BHOPS=999999
            BESTK=0
                  DO 57 K=1,NAVAIL
                  NHOPS=1
                  CHAN=CAVAIL(K)
561               LNODE=NODCHL(CHAN)
                  KNODE=LNODE-NSITES
                  IF(LINKTB(KNODE,J).GT.0)GO TO 562
                  NHOPS=NHOPS+1
                  CHAN=LINKTB(KNODE,INODE(KNODE,J))
                  GO TO 561
562               IF(NHOPS.GE.BHOPS)GO TO 57
                  BHOPS=NHOPS
                  BESTK=K
57                CONTINUE
            DSTALT(II,J)=CAVAIL(BESTK)
            DSTALT(II,JJ)=CAVAIL(BESTK)
            GO TO 59
58          DSTALT(II,J)=0
            DSTALT(II,JJ)=0
59          CONTINUE
60    CONTINUE
C
```

```
C PRINT OUT THE ROUTING TABLES.
      WRITE(6,3006)
3006  FORMAT(1HO,5X,'ROUTING TABLES ARE UPDATED NOW.')
      IF(IPASS.GT.1)GO TO 17
      NP1=NSITES+1
      WRITE(6,3003)
3003  FORMAT(1HO,5X,'PRIMARY ROUTING TABLE:')
      WRITE(6,3004)((DESTAB(I,J),J=1,NSITES),I=NP1,ALLDST)
3004  FORMAT(26I3)
      WRITE(6,3005)
3005  FORMAT(1HO,5X,'ALTERNATE ROUTING TABLE:')
      WRITE(6,3004)((DSTALT(I,J),J=1,NSITES),I=NP1,ALLDST)
C
C UPDATE THE PARM3 ARRAY TO REFLECT PROPER CAPACITIES IN SLOTS.
17    N=1
      DO 80 I=1,NM1
      L=I+1
          DO 79 J=L,NSITES
          IF(CONECT(I,J).EQ.O)GO TO 79
          PARM3(N)=NSLOTS(CONECT(I,J))
          PARM3(N+1)=PARM3(N)
          N=N+2
79        CONTINUE
80    CONTINUE
C
C NOTE:  CURRENT SUBPROGRAM SIMULA DIMENSION LIMITS ALLOW FOR:
C     (1)  PARAM(1).LE.52  (I.E., NSITES.LE.26)
C     (2)  PARAM(2).LE.160 (I.E., NLINKS.LE.80)
C     (3)  PARAM(3).LE.8000  (I.E., PARAM(3) IS NOW THE SUM OF
C                                  ALL PARM3 VALUES.)
C IF ANY OF THESE CONDITIONS ARE VIOLATED, THEN
C REDIMENSIONING IS NECESSARY.
      FLAG=O
      PARAM(3)=O
      ILIM=PARAM(2)
      DO 702 I=1,ILIM
      PARAM(3)=PARAM(3)+PARM3(I)
      JARM3(I)=PARAM(3)
702   CONTINUE
7000  FORMAT(1HO,5X,'DIMENSION CONDITION',I2,' IS VIOLATED.  ',
     1 'REDIMENSIONING IS REQUIRED FOR PROPER EXECUTION.')
      DO 703 I=1,3
      IF(PARAM(I).LE.DIMLIM(I))GO TO 703
      FLAG=1
      WRITE(6,7000)I
703   CONTINUE
704   IF(FLAG.EQ.O)GO TO 777
      STOP
777   RETURN
      END
C
C*****************************************************************
C
C SUBROUTINE SIMULA DRIVES THE NETWORK SIMULATION ROUTINES.
C THIS IS THE DRIVER-IT BUILDS USER DEFINED TABLES,
C INITIALIZES ACTIVITY AT EACH NODE, AND EMPLOYS A TIGHT
C DO LOOP, CALLING THE EVENT MODULE UNTIL THE RUN TIME
C SPECIFICATION IS EXCEEDED.  AT THIS POINT THE STATISTICS
C SUBROUTINE IS CALLED, FOLLOWED BY PROGRAM TERMINATION.
C
C*****************************************************************
C
      SUBROUTINE SIMULA(IPASS)
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
```

```
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
1003  FORMAT(8(I2,1X),2(I10,1X),2(I5,1X),I7,1X,I3,1X,I4,1X,I2)
1004  FORMAT(10I2)
1005  FORMAT(4(I5,1X))
1006  FORMAT(12I2)
1007  FORMAT(I10)
2000  FORMAT('1',40X,'SYSTEM PARAMETERS')
2001  FORMAT(' ',1X,3(I5,1X),I2,2X,I4,'MS ',I2,' MS',3X,I2,
     1'MIN',3X,I2,'SEC',2X,I5,' MS',I8,'MS ',I7,1X,
     2I5,'KBS ',I7,3X,I3,'SEC ',I4,'B',2X,I2)
2011  FORMAT('0',1X,'NODES LINKS SLOTS RATIO SLOT NODE  CS',6X,
     1'PS  MSG START TIME END TIME PACKET  VDR RATES ',
     2'Q SIZE   CS       PACKET PACKETS')
2012  FORMAT(' ',25X,'TIME DELAY ARRIVAL ARRIVAL',21X,
     1'LOADING',19X,'SERVICE SIZE   PER MSG')
3001  FORMAT(1HO,5X,'SEED TABLES:')
3002  FORMAT(4(1X,I15))
3003  FORMAT(1HO,5X,'PRIMARY ROUTING TABLE:')
3004  FORMAT(20(1X,I3))
3005  FORMAT(1HO,5X,'ALTERNATE ROUTING TABLE:')
3006  FORMAT(1HO,5X,'SOURCE, DESTINATION, AND CAPACITY (IN SLOTS) ',
     1 'FOR EACH CHANNEL:'//5X,'CHAN  SORCHL(I)  NODCHL(I)  ',
     2 'PARM3(I)')
3007  FORMAT(5X,I4,3(I7,4X))
C
      ALLDST=PARAM(1)
      NDEST=ALLDST/2
      PARAM(17)=0
      NCHNLS=PARAM(2)
C INIT AVAIL SLOTS PER CHANNEL TABLE
77    DO 80 I=1,NCHNLS
      NLINES(I)=PARM3(I)
80    CONTINUE
C     WRITE(6,2000)
C     WRITE(6,2011)
C     WRITE(6,2012)
C     WRITE(6,2001)(PARAM(I),I=1,16)
C     WRITE(6,3001)
C     WRITE(6,3002)((SEEDTB(I,J),J=1,4),I=1,ALLDST)
C     WRITE(6,3003)
C     WRITE(6,3004)((DESTAB(I,J),J=1,ALLDST),I=1,ALLDST)
C     WRITE(6,3005)
C     WRITE(6,3004)((DSTALT(I,J),J=1,ALLDST),I=1,ALLDST)
C     WRITE(6,3006)
C     DO 81 I=1,NCHNLS
C     WRITE(6,3007)I,SORCHL(I),NODCHL(I),PARM3(I)
C81   CONTINUE
      CALL BDATA
      CLASS=2
C CREATE AN EVENT TABLE ENTRY FOR EACH NODE.
      DO 10 I=1,ALLDST
      IF(I.GT.NDEST)CLASS=1
      CALL NEWMSG(I,CLASS)
      CALL NUEVNT(I,CLASS)
10    CONTINUE
C START THE SIMULATION.
20    CALL EVENT
C IF TIME LIMIT UP GO PRINT STATISTICS.
      IF(PARAM(9).LT.PARAM(10)) GO TO 20
      PARAM(10)=PARAM(9)
      PARAM(9)=PARAM(17)
      CALL STATX
C     CALL STATS
C STOP THE SIMULATION.
999   RETURN
```

```
      END
      SUBROUTINE BDATA
C
C*****************************************************************
C
C THE DATA BLOCK MAKES IT CONVENIENT FOR THE USER
C TO INITIALIZE HIS STORAGE AREAS.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
C
      DO 10 I=1,52
      QCNT(I)=0
          DO 9 J=1,52
          LINKTB(I,J)=0
9         CONTINUE
          DO 8 J=1,5
          EVTBL(I,J)=0
8         CONTINUE
10    CONTINUE
C
      DO 20 I=1,26
      APCKTS(I)=0
      TDELAY(I)=0.0
          DO 15 J=1,26
          DSTLOD(I,J)=0
          DSTCNT(I,J)=0
          CUMLOD(I,J)=0
          CUMCNT(I,J)=0
15        CONTINUE
          DO 16 J=1,3
          CALLS(I,J)=0
          CSARV(I,J)=0
          NODLOD(I,J)=0
16        CONTINUE
          DO 17 J=1,13
          CUMTIM(I,J)=0
17        CONTINUE
          DO 18 J=1,200
          CALLQ(I,J)=0
18        CONTINUE
          DO 19 J=1,1800
          QUEUE(I,J)=0
19        CONTINUE
20    CONTINUE
C
      DO 30 I=1,8000
      DO 30 J=1,11
      CHANTB(I,J)=0
30    CONTINUE
C
      DO 40 I=1,4
      EVENTX(I)=0
40    CONTINUE
C
      DO 50 I=1,160
      ALTCH(I)=0
      ROUT(I)=0
50    CONTINUE
```

```
      RETURN
      END
      SUBROUTINE NEWMSG(NODE,CLASS)
C
C*****************************************************************
C
C THIS ROUTINE GENERATES VOICE AND PACKET ARRIVALS.
C INFORMATION RELATING TO EACH ARRIVAL RESULTS IN
C A QUEUE ENTRY BEING BUILT.  IF THE CURRENT LOAD EXCEEDS
C A USER SPECIFIED STEADY-STATE LOAD, CHANNEL TABLE
C STATISTICAL GATHERING ENTRIES ARE ZEROED OUT.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      REAL*4 ALOG
      REAL*4 RN
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      DATA FLAG/1/
C KNODE IS USED WITH CIRCUIT SWITCH NODES TO ALLOW
C PROPER SUBSCRIPTING TO OCCUR OF THE CIRCUIT SWITCH TABLES.
      IF((CLASS.EQ.1).AND.(PARAM(7).EQ.0))GO TO 170
      IF((CLASS.EQ.2).AND.(PARAM(8).EQ.0))GO TO 170
      IF(CLASS.EQ.1)KNODE=NODE-(PARAM(1)/2)
      IY=0
      RN=0.0
      STIME=PARAM(9)
10    IX=SEEDTB(NODE,3)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,3)=IY
      DEST=PARAM(1)*RN+1
C KEEP GENERATING A NEW NODE UNTIL ONE DIFFERENT FROM
C THE SOURCE IS FOUND.
      IF(DEST.EQ.NODE)GO TO 10
      IF(CLASS.EQ.2)GO TO 20
C THIS IS A CS DEST
      IF(DEST.LE.(PARAM(1)/2))GO TO 10
      CSARV(KNODE,3)=DEST
      IX=SEEDTB(NODE,1)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,1)=IY
C GENERATE ARRIVAL TIME FOR CIRCUIT SWITCH TRANSACTION.
      ARV=(60000*(-1.0/PARAM(7)*ALOG(RN)))+PARAM(9)
      IX=SEEDTB(NODE,2)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,2)=IY
      DEP=-1000.0*PARAM(14)*ALOG(RN)+ARV
      CSARV(KNODE,1)=ARV
      CSARV(KNODE,2)=DEP
100   QCNT(NODE)=QCNT(NODE)+1
170   RETURN
20    IF(DEST.GT.(PARAM(1)/2))GO TO 10
      X=0.0
      XLAMDA=PARAM(8)
      CALL POISSN(XLAMDA,X,NODE)
      NMSGS=X
      IX=SEEDTB(NODE,1)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,1)=IY
      KEY=0
      LIMIT=QCNT(NODE)
```

```
      ITOP=PARAM(13)-4
      DO 40 I=2,ITOP,6
      IF(QUEUE(NODE,(I-1)).EQ.0)GO TO 40
      IF(QUEUE(NODE,(I-1)).EQ.100)GO TO 46
      IF(QUEUE(NODE,(I-1)).EQ.9999999)GO TO 46
      IF(QUEUE(NODE,I).LE.KEY)GO TO 40
      KEY=QUEUE(NODE,I)
46    LIMIT=LIMIT-1
      IF(LIMIT.EQ.0)GO TO 50
40    CONTINUE
50    IF(KEY.GT.STIME)STIME=KEY
C GENERATE ARRIVAL TIME FOR DATA TRAFFIC.
      ARV=(1000*(-1.0*ALOG(RN)))+STIME
C NOW GENERATE NUM OF PACKETS
      XPROB=PARAM(16)/100.0
      LENGTH=0
      NUM=0
      IF(NMSGS.EQ.0)NMSGS=1
      DO 25 K=1,NMSGS
      CALL GEOM(XPROB,NUM,NODE)
      LENGTH=LENGTH+NUM
25    CONTINUE
      IF(LENGTH.EQ.0)LENGTH=1
      DEP=LENGTH*PARAM(5)+ARV
      ITOP=PARAM(13)
      DO 30 I=1,ITOP,6
      IF(QUEUE(NODE,I).NE.0)GO TO 30
C BUILD UP THE DATA QUEUE ENTRY.
      QUEUE(NODE,I)=1
      QUEUE(NODE,I+1)=ARV
      QUEUE(NODE,I+2)=DEP
      QUEUE(NODE,I+3)=LENGTH
      QUEUE(NODE,I+4)=DEST
      QUEUE(NODE,I+5)=NMSGS
C UPDATE NODE COUNTERS.
      DSTCNT(NODE,DEST)=DSTCNT(NODE,DEST)+NMSGS
      DSTLOD(NODE,DEST)=DSTLOD(NODE,DEST)+LENGTH
      NODLOD(NODE,1)=NODLOD(NODE,1)+LENGTH
      NODLOD(NODE,2)=NODLOD(NODE,2)+LENGTH
      NODLOD(NODE,3)=NODLOD(NODE,3)+NMSGS
      CUMLOD(NODE,DEST)=CUMLOD(NODE,DEST)+LENGTH
      CUMCNT(NODE,DEST)=CUMCNT(NODE,DEST)+NMSGS
220   IF(I.GT.(PARAM(13)-12))GO TO 160
      IF(FLAG.EQ.0)GO TO 100
C WE HAVE REACHED STEADY-STATE.  MUST INITIALIZE COUNTERS.
      IF(NODLOD(NODE,1).GE.PARAM(11))GO TO 110
      GO TO 100
30    CONTINUE
      GO TO 100
110   FLAG=0
      NDEST=PARAM(1)/2
      ALLDST=PARAM(1)
      ITOP=PARAM(3)
      DO 130 I=1,ITOP
      CHANTB(I,6)=0
      CHANTB(I,9)=0
      CHANTB(I,10)=0
      CHANTB(I,11)=0
130   CONTINUE
      DO 140 I=1,NDEST
      DO 140 J=1,13
      CUMTIM(I,J)=0
140   CONTINUE
      DO 150 I=1,NDEST
      DO 150 J=1,2
      CALLS(I,J)=0
150   CONTINUE
      PARAM(17)=PARAM(9)
```

```
      GO TO 100
160   WRITE(6,2001)NODE
      PARAM(10)=PARAM(9)
      PARAM(9)=0
      CALL STATX
C     CALL STATS
      STOP
1003  FORMAT(' ',11(I9,1X))
1005  FORMAT('0','CHANNEL ',I2)
2001  FORMAT('0','QUEUE SIZE EXCEEDED-NEWMSG 30-NODE=',I4)
2002  FORMAT(' ',6(I9,1X))
2003  FORMAT(' ',5(I6,1X))
      END
      SUBROUTINE POISSN(XLAMDA,X,NODE)
C
C*****************************************************************
C
C THIS IS THE POISSON ARRIVAL GENERATOR.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      REAL*4 RN
      REAL*4 EXP
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      X=0.0
      T=EXP(-XLAMDA)
      T1=1.0
      IY=0
4     IX=SEEDTB(NODE,2)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,2)=IY
      T1=T1*RN
      IF(T1-T)9,7,7
7     X=X+1.0
      GO TO 4
9     RETURN
      END
      SUBROUTINE GEOM(XPROB,NUM,NODE)
C
C*****************************************************************
C
C THIS ROUTINE GENERATES THE NUMBER OF PACKETS
C FOR A DATA TRANSACTION.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      REAL*4 ALOG10
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      REAL*4 RN
      NUM=0
      IY=0
10    IX=SEEDTB(NODE,4)
```

```
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,4)=IY
      IF(RN.LE.XPROB)GO TO 20
      NUM=NUM+1
      GO TO 10
20    RETURN
      END
      SUBROUTINE RANDOM(IX,IY,RN)
C
C******************************************************************
C
C THIS IS THE RANDOM NUMBER GENERATOR.
C
C******************************************************************
C
      IMPLICIT INTEGER (A-Q)
      IY=IX*65539
      IF(IY)3,4,4
3     IY=IY+2147483647+1
4     RN=IY
      RN=RN*.4656613E-9
      IX=IY
      RETURN
      END
      SUBROUTINE NUEVNT(NODE,CLASS)
C
C******************************************************************
C
C THIS ROUTINE IS RESPONSIBLE FOR SELECTING THE NEXT
C ACTIVITY AT A NODE.  ONCE AN EVENT IS SELECTED
C (ARRIVAL OR DEPARTURE), INFORMATION PERTAINING TO IT
C IS PLACED IN THE EVENT TABLE ENTRY FOR THAT NODE.
C
C******************************************************************
C
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      IDELAY=0
      IYESNO=0
      LIMIT=QCNT(NODE)
C DETERMINE IF THIS IS A PACKET OR CIRCUIT NODE.
      IF(CLASS.EQ.2)GO TO 110
C MUST GET NEW CS ENTRY
      KNODE=NODE-(PARAM(1)/2)
C SET KEY TO MAX VALUE.
      DEP=9999999
C NOW SEARCH FOR AN EVENT SMALLER THAN THE KEY.
C SEARCH CALLQ TABLE LOOKING FOR SOONEST ACTIVITY.
      DO 20 I=1,150,4
      IF(CALLQ(KNODE,I).EQ.0)GO TO 20
      IF(CALLQ(KNODE,I).GE.DEP)GO TO 15
      DEP=CALLQ(KNODE,I)
      INDEX=I
15    LIMIT=LIMIT-1
      IF(LIMIT.EQ.0)GO TO 5
20    CONTINUE
5     IF(DEP.EQ.9999999)GO TO 25
      IF(CSARV(KNODE,1).GE.DEP)GO TO 30
C NOW PLACE CIRCUIT SWITCH INFORMATION IN EVENT TABLE.
25    EVTBL(NODE,1)=CSARV(KNODE,1)
      EVTBL(NODE,2)=3
```

```
      EVTBL(NODE,3)=CSARV(KNODE,2)
      EVTBL(NODE,4)=CSARV(KNODE,3)
      GO TO 100
30    EVTBL(NODE,1)=DEP
      EVTBL(NODE,2)=4
      EVTBL(NODE,4)=CALLQ(KNODE,(INDEX+1))
      EVTBL(NODE,5)=INDEX
      GO TO 100
110   IF(PARAM(8).EQ.0)GO TO 100
      KEY=9999999
      ITOP=PARAM(13)-3
      DO 40 I=3,ITOP,6
C IF QUEUE(1) EQUAL ZERO, THEN SKIP IT.
C IF QUEUE(1) EQUAL 100, SKIP IT BECAUSE IT IS THE
C RETURN PATH FOR A CONNECTION.
      IF(QUEUE(NODE,(I-2)).EQ.0)GO TO 40
      IF(QUEUE(NODE,(I-2)).EQ.100)GO TO 40
      IF(QUEUE(NODE,I).GE.KEY)GO TO 41
      IF(QUEUE(NODE,(I-2)).NE.9999999)GO TO 41
      KEY=QUEUE(NODE,I)
      FLAG=2
      INDEX=I-2
41    LIMIT=LIMIT-1
      IF(LIMIT.EQ.0)GO TO 45
40    CONTINUE
C THIS LOGIC CHECKS TRAFFIC THAT HAS BEEN ON QUEUE FOR
C SOME TIME TO SEE IF IT CAN BE SENT YET.
45    PRIORY=0
      KK=6
      DO 55 KKK=1,5
      K=KK-KKK
      LIMIT=QCNT(NODE)
      IF(PRIORY.NE.0)GO TO 75
      ITOP=PARAM(13)-4
      DO 50 I=2,ITOP,6
      IF(QUEUE(NODE,(I-1)).EQ.0)GO TO 50
      IF(QUEUE(NODE,(I-1)).EQ.9999999)GO TO 46
      IF(QUEUE(NODE,(I-1)).EQ.100)GO TO 46
      IF(QUEUE(NODE,I).GE.KEY)GO TO 46
      IF(QUEUE(NODE,(I-1)).NE.K)GO TO 46
      DEST=QUEUE(NODE,(I+3))
C HAVE FOUND THE OLDEST TRAFFIC, CHECK FIRST PATH.
      PASS=1
      CALL ROUTE(NODE,DEST,IYESNO,IDELAY,CLASS,PASS)
      IF(IYESNO.EQ.1)GO TO 46
      IF(IYESNO.EQ.2)GO TO 46
C NOW CHECK RETURN PATH.  IF OK THEN IT CAN BE SENT.
      PASS=2
      CALL ROUTE(DEST,NODE,IYESNO,IDELAY,CLASS,PASS)
      IF(IYESNO.EQ.2)GO TO 46
      IF(IYESNO.EQ.1)GO TO 46
80    KEY=QUEUE(NODE,I)
      FLAG=1
      INDEX=I-1
      PRIORY=K
46    LIMIT=LIMIT-1
      IF(LIMIT.LE.0)GO TO 55
50    CONTINUE
55    CONTINUE
      IF(KEY.NE.9999999)GO TO 75
      IF(KKK.LT.5)GO TO 75
      LIMIT=QCNT(NODE)
      FLAG=1
      ITOP=PARAM(13)-5
      KEY=QUEUE(NODE,2)
      DO 35 I=1,ITOP,6
C GET SOONEST DEPARTURE.
      IF(QUEUE(NODE,I).EQ.0)GO TO 35
```

```
      IF(QUEUE(NODE,I).LT.6)GO TO 36
35    CONTINUE
36    KEY=QUEUE(NODE,(I+1))
      INDEX=I
      DO 60 I=1,ITOP,6
C GET SOONEST ARRIVAL
      IF(QUEUE(NODE,I).EQ.0)GO TO 60
      IF(QUEUE(NODE,I).EQ.100)GO TO 65
      IF(QUEUE(NODE,I).EQ.9999999)GO TO 65
      IF(QUEUE(NODE,(I+1)).GT.KEY)GO TO 85
86    IF(QUEUE(NODE,I).LT.5)GO TO 65
      QUEUE(NODE,I)=1
65    LIMIT=LIMIT-1
      IF(LIMIT.EQ.0)GO TO 75
      GO TO 60
85    KEY=QUEUE(NODE,(I+1))
      INDEX=I
      IF(QUEUE(NODE,(I+1)).LE.PARAM(9))GO TO 86
      GO TO 75
60    CONTINUE
C SELECT FROM ARRIVAL OR DEPARTURE, AND PLACE INOFRMATION
C RELATING TO IT ON EVENT TABLE.
75    EVTBL(NODE,1)=QUEUE(NODE,(INDEX+1))
      IF(FLAG.EQ.2)EVTBL(NODE,1)=QUEUE(NODE,(INDEX+2))
      EVTBL(NODE,2)=FLAG
      EVTBL(NODE,3)=QUEUE(NODE,(INDEX+3))
      EVTBL(NODE,4)=QUEUE(NODE,(INDEX+4))
      EVTBL(NODE,5)=INDEX
100   RETURN
      END
      SUBROUTINE STATS
C
C*****************************************************************
C
C THIS ROUTINE IS SELF DOCUMENTING.
C IT IS RESPONSIBLE FOR OUTPUT GENERATION OF STATISTICAL
C INFORMATION.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
C WRITE FINAL EVTBL INFORMATION.
      WRITE(6,2020)((EVTBL(J,K),J=1,5),K=1,5)
C WRITE OUT PACKET NODE INFORMATION.
      WRITE(6,2005)
      WRITE(6,2006)
      ITOP=PARAM(1)/2
      DO 30 I=1,ITOP
      WRITE(6,2007)I,(CUMTIM(I,K),K=1,13)
30    CONTINUE
      WRITE(6,2013)
      IUP=PARAM(13)-4
      DO 60 I=1,ITOP
      DO 90 J=2,IUP,6
      IF(QUEUE(I,J).LE.PARAM(10))GO TO 90
      JPTR=QUEUE(I,J+3)
      NODLOD(I,1)=NODLOD(I,1)-10
      DSTLOD(I,JPTR)=DSTLOD(I,JPTR)-10
      DSTCNT(I,JPTR)=DSTCNT(I,JPTR)-1
90    CONTINUE
```

```
      WRITE(6,2014)I,(NODLOD(I,K),K=1,3)
60    CONTINUE
      WRITE(6,2015)
      ALLDST=PARAM(1)/2
      DO 70 I=1,ALLDST
      DO 70 J=1,ALLDST
      IF(I.EQ.J)GO TO 70
      WRITE(6,2016)I,J,DSTLOD(I,J),DSTCNT(I,J),CUMLOD(I,J),CUMCNT(I,J)
70    CONTINUE
      RETURN
2002  FORMAT('-',40X,'CHANNEL',2X,I2,2X,'UTILIZATION')
2003  FORMAT(' ','SLOT',5X,'PACKETS SENT',6X,'PER CENT UTILIZATION',
     15X,'NUMBER OF VOICE CALLS')
2004  FORMAT(' ',2X,I2,9X,I8,22X,F4.2,20X,I6)
2005  FORMAT('1',40X,'PACKET NODE STATISTICS')
2006  FORMAT('O','NODE',5X,'DELAY(SECS)',4X,'<.1',4X,'<.2',4X,
     1'<.3',4X,'<.4',4X,'<.5',4X,'<.6',4X,'<.7',4X,'<.8',4X,
     2'<.9',4X,'< 1',4X,'< 2',4X,'< 5',4X,'> 5')
2007  FORMAT(' ',2X,I2,14X,I9,11I7,I9)
2008  FORMAT('O',40X,'CIRCUIT SWITCH NODE STATISTICS')
2009  FORMAT('O','NODE',5X,'TOTAL CALLS',5X,'CALLS LOST',9X,
     1'BLOCKING',5X,'CALLS IN SYSTEM')
2010  FORMAT(' ',2X,I2,11X,I5,10X,I5,13X,F4.2,10X,I5)
2013  FORMAT('-','NODE CURRENT PACKET LOAD CUMULATIVE PACKET ',
     1'LOAD CUMULATIVE TRANSACTIONS')
2014  FORMAT(' ',2X,I2,10X,I10,13X,I10,14X,I10)
2015  FORMAT('-','NODE DEST CURRENT PACKET LOAD CURRENT ',
     1'TRANSACTION LOAD CUMULATIVE PACKET LOAD CUMULATIVE',
     2' TRANSACTION LOAD')
2016  FORMAT(' ',2X,I2,3X,I2,10X,I10,15X,I10,13X,I10,18X,I10)
2017  FORMAT('O','TOTAL PACKETS SENT ',I7,' TOTAL LINK UTILIZATION',
     12X,F4.2)
2020  FORMAT(' ',5(I6,1X))
      END
      SUBROUTINE EVENT
C
C*****************************************************************
C
C THIS ROUTINE SCANS THE EVENT TABLE ENTRIES LOOKING FOR
C THE NEXT SYSTEM EVENT.  IT THEN BRANCHES TO THE
C APPROPRIATE ROUTINE TO SERVICE THE EVENT.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      NOROUT=0
      BEST=9999999
C FIND SOONEST ACTIVITY
      ALLDST=PARAM(1)
      DO 10 I=1,ALLDST
      IF(EVTBL(I,1).EQ.O)GO TO 10
      IF(EVTBL(I,1).GE.BEST)GO TO 10
      BEST=EVTBL(I,1)
      NODE=I
10    CONTINUE
C NOW PROCESS EVENT OCCURRENCE TYPE
C 1=PS ARV, 2=PS DEP, 3=CS ARR, 4=CS DEP
      KEY=EVTBL(NODE,2)
      EVENTX(KEY)=EVENTX(KEY)+1
      GO TO (100,200,300,400),KEY
```

```
C PROCESS PS ARRIVAL
100   CLASS=2
      CALL ARRIVE(NODE,CLASS)
150   CALL NEWMSG(NODE,CLASS)
155   CALL NUEVNT(NODE,CLASS)
160   RETURN
C PROCESS PS DEPARTURE
200   CLASS=2
      CALL DEPART(NODE,CLASS)
      GO TO 155
C NOW PROCESS CS ARRIVAL
300   CLASS=1
      CALL ARRIVE(NODE,CLASS)
      GO TO 150
C CS DEPARTURE
400   CLASS=1
      CALL DEPART(NODE,CLASS)
      GO TO 155
      END
      SUBROUTINE ROUTE(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
C
C*****************************************************************
C
C THIS ROUTINE IS USED TO FIND A ROUTE THROUGH THE NETWORK.
C IT IS CALLED TWICE FOR EACH CIRCUIT SWITCH ROUTE.  NTRACE IS
C A TABLE USED TO INSURE NO LOOPS OCCUR IN THE ROUTING
C PROCESS.  IYESNO SIGNIFIES WHETHER OR NOT A GOOD
C ROUTE WAS FOUND.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      DIMENSION NTRACE(160)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      IROUT=PARAM(2)
      IF(PASS.EQ.2)GO TO 80
      DO 85 I=1,IROUT
      ALTCH(I)=0
      ROUT(I)=0
85    CONTINUE
80    DO 90 I=1,IROUT
      NTRACE(I)=0
90    CONTINUE
      NODE=LNODE
      DEST=KDEST
      INODE=NODE
      NCOUNT=1
C
      ITYPE=0
      IDELAY=0
      RATIO=1
      ALT=0
      IF(CLASS.EQ.2)RATIO=PARAM(4)
      NTRACE(NCOUNT)=INODE
      IYESNO=0
C IS THIS A PS ROUTE
      IF(CLASS.EQ.2)INODE=DESTAB(NODE,DEST)
      IF(CLASS.EQ.1)GO TO 10
      IDELAY=PARAM(6)
      NCOUNT=NCOUNT+1
      NTRACE(NCOUNT)=INODE
```

```
C DOES PATH ALREADY EXIST FOR THIS TRANSACTION
      ICHAN=LINKTB(INODE,DEST)
      IF(ICHAN.EQ.O)GO TO 10
      ITYPE=1
10    IF(INODE.EQ.DEST)GO TO 60
      ICHAN=DESTAB(INODE,DEST)
25    IF(RATIO.GT.NLINES(ICHAN))GO TO 20
C THERE ARE SLOTS AVAILABLE
      ROUT(ICHAN)=RATIO+ROUT(ICHAN)
      IF((NLINES(ICHAN)-ROUT(ICHAN)).LT.O)GO TO 50
      ALT=O
      INODE=NODCHL(ICHAN)
      DO 30 K=1,NCOUNT
      IF(INODE.EQ.NTRACE(K))GO TO 50
30    CONTINUE
      NCOUNT=NCOUNT+1
      NTRACE(NCOUNT)=INODE
C ADD DELAY TIME FOR SIGNALLING THROUGH THIS NODE.
      IDELAY=PARAM(6)+IDELAY
      IF(DESTAB(INODE,DEST).EQ.O)GO TO 60
      GO TO 10
20    IF(ALT.EQ.1)GO TO 50
      ALT=1
      IF(ALTCH(ICHAN).EQ.1)GO TO 50
      ALTCH(ICHAN)=1
      ICHAN=DSTALT(INODE,DEST)
      GO TO 25
50    IYESNO=1
60    IF(ITYPE.EQ.IYESNO)GO TO 70
      IF(ITYPE.EQ.O)IYESNO=2
      IF(ITYPE.EQ.1)IYESNO=3
70    RETURN
      END
      SUBROUTINE DEPART(LNODE,CLASS)
C
C*******************************************************************
C
C THIS ROUTINE IS THE DRIVER FOR DATA/VOICE TRANSACTION
C TERMINATIONS.  IT FINDS THE CHANNEL ENTRY TO START THE
C REMOVAL PROCESS.  IT THEN CALLS REMOVE TO ACTUALLY PURGE
C TABLE ENTRIES.  TRAFFIC LOAD STATISTICS ARE THEN
C UPDATED BY THIS MODULE.
C
C*******************************************************************
C
      IMPLICIT INTEGER (A-S)
      DIMENSION INDEX(2)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      NODE=LNODE
      DEST=EVTBL(NODE,4)
70    QADDR=O
      CHPTR=O
      PASS=1
C GO FIND STARTING POINT FOR REMOVAL OF THIS TRANSACTION.
      CALL GETQ(NODE,DEST,CLASS,QADDR,CHPTR,PASS)
      IF(QADDR.FQ.O)GO TO 50
      INDEX(1)=QADDR
      PASS=2
      CALL GETQ(DEST,NODE,CLASS,QADDR,CHPTR,PASS)
      IF(QADDR.EQ.O)GO TO 50
      INDEX(2)=QADDR
```

```
C
C REMOVE FIRST HALF OF CONNECTION.
      PASS=1
      CALL REMOVE(NODE,DEST,CLASS,PASS)
C REMOVE RETURN PATH FOR TRANSACTION.
      PASS=2
      CALL REMOVE(DEST,NODE,CLASS,PASS)
C DETERMINE IF TRANSACTION WAS A CIRCUIT OR DATA TRANSACTION.
      IF(CLASS.EQ.1)GO TO 20
      KNODE=NODE
      KDEST=DEST
      DO 10 I=1,2
      DEST=KDEST
      NODE=KNODE
      QCNT(NODE)=QCNT(NODE)-1
      K=INDEX(I)
C UPDATE PACKET NODE STATISTICS.
      IF(DSTLOD(NODE,DEST)-QUEUE(NODE,(K+3)).LT.0)GO TO 30
      DSTLOD(NODE,DEST)=DSTLOD(NODE,DEST)-QUEUE(NODE,(K+3))
      DSTCNT(NODE,DEST)=DSTCNT(NODE,DEST)-QUEUE(NODE,(K+5))
      NODLOD(NODE,1)=NODLOD(NODE,1)-QUEUE(NODE,(K+3))
30    ITOP=K+5
C PURGE THE Q ENTRIES FOR DATA.
      DO 40 M=K,ITOP
      QUEUE(NODE,M)=0
40    CONTINUE
      KNODE=DEST
      KDEST=NODE
10    CONTINUE
35    PARAM(9)=EVTBL(LNODE,1)
80    RETURN
20    KNODE=NODE-(PARAM(1)/2)
      KDEST=DEST-(PARAM(1)/2)
C REMOVE CIRCUIT SWITCH TRANSACTION FROM CALLQ TABLE.
      CALLQ(KNODE,(INDEX(1)))=0
      CALLQ(KDEST,(INDEX(2)))=0
      KNODE=CALLQ(KDEST,(INDEX(2)+3))-(PARAM(1)/2)
      CALLS(KNODE,3)=CALLS(KNODE,3)-1
      CALL NUEVNT(DEST,CLASS)
      GO TO 35
50    K=EVTBL(LNODE,5)
      ITOP=K+5
      DO 60 M=K,ITOP
      QUEUE(LNODE,M)=0
60    CONTINUE
      GO TO 80
      END
      SUBROUTINE REMOVE(LNODE,KDEST,CLASS,PASS)
C
C*******************************************************************
C
C THIS ROUTINE RIPPLES THROUGH ALL CHANNEL ENTRIES,
C ZEROING OUT ENTRIES PERTAINING TO THIS ROUTE.
C
C*******************************************************************
C
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      NODE=LNODE
      DEST=KDEST
      IF(CLASS.EQ.1)KNODE=NODE-(PARAM(1)/2)
```

```
      FLAG=1
      ILIM=1
      IF(CLASS.EQ.2)ILIM=PARAM(4)
      INODE=NODE
      IF(CLASS.EQ.2)INODE=DESTAB(NODE,DEST)
      ICHAN=DESTAB(INODE,DEST)
10    IF(INODE.EQ.DEST)GO TO 40
      FLAG=1
35    IF(FLAG.EQ.O)ICHAN=DSTALT(INODE,DEST)
      JCHNL=JARM3(ICHAN)-PARM3(ICHAN)+1
      ITOP=JCHNL+PARM3(ICHAN)-1
      I=JCHNL-1
      DO 15 J=1,ILIM
      JCHNL=I+1
C
      DO 20 I=JCHNL,ITOP
      IF(CHANTB(I,7).NE.NODE)GO TO 20
C CHECK FOR MATCH BY SOURCE, DEST, AND TIME.
      IF(CHANTB(I,1).NE.DEST)GO TO 20
      IF(PASS.EQ.2)GO TO 90
      IF(CHANTB(I,3).NE.EVTBL(NODE,1))GO TO 20
C HAVE FOUND CHANNEL MATCH, PURGE CHANNEL ENTRIES.
60    CHANTB(I,6)=CHANTB(I,6)+(CHANTB(I,3)-CHANTB(I,2))
      CHANTB(I,2)=O
      CHANTB(I,3)=O
      CHANTB(I,1)=O
      NLINES(ICHAN)=NLINES(ICHAN)+1
      CHANTB(I,4)=O
      INODE=CHANTB(I,5)
      CHANTB(I,5)=O
      GO TO 70
90    IF(CHANTB(I,3).EQ.EVTBL(DEST,1))GO TO 60
20    CONTINUE
      IF(FLAG.EQ.O)GO TO 80
      FLAG=O
      GO TO 35
70    IF(CLASS.EQ.1)CHANTB(I,11)=CHANTB(I,11)+1
      IF(CLASS.EQ.1)GO TO 15
      PTR=CHANTB(I,8)
      CHANTB(I,9)=CHANTB(I,9)+QUEUE(NODE,(PTR+5))
      CHANTB(I,10)=CHANTB(I,10)+QUEUE(NODE,(PTR+3))
15    CONTINUE
C KEEP LOOPING THROUGH THIS CHANNEL TABLE UNTIL ALL ENTRIES
C ARE FOUND AND PURGED.
30    ICHAN=DESTAB(INODE,DEST)
      IF(DESTAB(INODE,DEST).EQ.O)GO TO 40
      GO TO 10
80    WRITE(6,1000)LNODE,KDEST
40    IF(CLASS.NE.2)GO TO 50
      INODE=DESTAB(NODE,DEST)
50    RETURN
1000  FORMAT('O','ERROR IN REMOVE',2(1X,I2))
      END
      SUBROUTINE ARRIVE(LNODE,CLASS)
C
C*****************************************************************
C
C THIS ROUTINE IS RESPONSIBLE FOR ARRIVAL OF A DATA/VOICE
C TRANSACTION AT THIS NODE.  IT IS THE DRIVER-WITH RESPONSIBILITY
C FOR GETTING A ROUTE, UPDATING CHANNEL TABLES, AND
C GENERATING PACKET DELAY INFORMATION.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      DIMENSION LDELAY(12)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
```

```
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      DATA LDELAY/101,201,301,401,501,601,701,801,901,1001,2001,5001/
      IDELAY=0
      FLAG=1
      NODE=LNODE
      IF(EVTBL(NODE,1).GT.PARAM(9)) PARAM(9)=EVTBL(NODE,1)
      STIME=PARAM(9)
      DEST=EVTBL(NODE,4)
      INDXQ=EVTBL(NODE,5)
      IYESNO=0
      PASS=1
      CALL ROUTE(NODE,DEST,IYESNO,IDELAY,CLASS,PASS)
C BRANCH IF CS TRANSACTION
      IF(CLASS.EQ.1)GO TO 30
C VALUE OF IYESNO  0-NO EXISTING ROUTE, PATH AVAILABLE
C                  1-EXISTING ROUTE AVAILABLE, NO PATH
C                  2-NO EXISTING ROUTE AVAILABLE, NO PATH
C                  3-EXISTING PATH AVAILABLE, PATH AVAILABLE
      IF(IYESNO.EQ.2)GO TO 10
      IF(IYESNO.EQ.3)GO TO 20
      IF(IYESNO.EQ.0)GO TO 20
      IF(IYESNO.EQ.1)GO TO 10
C NOW ADD DELAY TO CUM TIME TABLE
75    IF(FLAG.EQ.1)GO TO 80
      NODE=DEST
80    APCKTS(NODE)=APCKTS(NODE)+NPCKTS
      TDELAY(NODE)=TDELAY(NODE)+IDELAY*NPCKTS/1000.
      DO 70 I=1,12
      IF(IDELAY.GT.LDELAY(I))GO TO 70
C ADD DELAY TIME FOR THESE PACKETS.
      CUMTIM(NODE,I)=CUMTIM(NODE,I)+NPCKTS
      IF(FLAG.EQ.0)GO TO 180
      FLAG=0
      NPCKTS=NPCKTS/5+1
      IDELAY=50
      GO TO 75
70    CONTINUE
      CUMTIM(NODE,13)=CUMTIM(NODE,13)+NPCKTS
      IF(FLAG.EQ.0)GO TO 180
      FLAG=0
      GO TO 75
180   LTOP=PARAM(2)
      DO 50 L=1,LTOP
      ALTCH(L)=0
50    CONTINUE
      RETURN
C NO PATH AVAILABLE
10    INDEX=EVTBL(NODE,5)
      QUEUE(NODE,INDEX)=QUEUE(NODE,INDEX)+1
      GO TO 180
C MUST BUILD NEW PS PATH
20    IDELAY=0
C NOW CHK 2ND HALF OF FDX LINE
      PASS=2
      CALL ROUTE(DEST,NODE,IYESNO,IDELAY,CLASS,PASS)
      IF(IYESNO.EQ.2)GO TO 10
      IF(IYESNO.EQ.1)GO TO 10
      IF(EVTBL(NODE,1).GE.STIME)GO TO 25
      IDELAY=IDELAY+STIME-EVTBL(NODE,1)
25    INDEX=EVTBL(NODE,5)
      NPCKTS=QUEUE(NODE,(INDEX+3))
      NMSGS=QUEUE(NODE,(INDEX+5))
C UPDATE QUEUE ENTRIES WITH INFORMATION PLACED IN CHANNEL TABLES.
```

```
      QUEUE(NODE,INDEX)=9999999
      QUEUE(NODE,(INDEX+1))=QUEUE(NODE,(INDEX+1))+IDELAY
      QUEUE(NODE,(INDEX+2))=QUEUE(NODE,(INDEX+2))+IDELAY
      ITOP=PARAM(13)
      DO 90 I=1,ITOP,6
C FIND A FREE QUEUE ENTRY FOR RETURN PATH.
      IF(QUEUE(DEST,I).NE.0)GO TO 90
      QUEUE(DEST,I)=100
      QUEUE(DEST,(I+1))=QUEUE(NODE,(INDEX+1))
      QUEUE(DEST,(I+2))=QUEUE(NODE,(INDEX+2))
      QUEUE(DEST,(I+3))=NPCKTS/5+1
      QUEUE(DEST,(I+4))=NODE
C ASSUME RETURN PATH CARRIES SOME TRAFFIC. ADD THIS FACTOR TO TABLES.
      QUEUE(DEST,(I+5))=NMSGS/5+1
      QCNT(DEST)=QCNT(DEST)+1
      DSTLOD(DEST,NODE)=DSTLOD(DEST,NODE)+NPCKTS/5+1
      DSTCNT(DEST,NODE)=DSTCNT(DEST,NODE)+NMSGS/5+1
      NODLOD(DEST,1)=NODLOD(DEST,1)+NPCKTS/5+1
      NODLOD(DEST,2)=NODLOD(DEST,2)+NPCKTS/5+1
      NODLOD(DEST,3)=NODLOD(DEST,3)+NMSGS/5+1
      CUMLOD(DEST,NODE)=CUMLOD(DEST,NODE)+NPCKTS/5+1
      CUMCNT(DEST,NODE)=CUMCNT(DEST,NODE)+NMSGS/5+1
      GO TO 35
90    CONTINUE
C GO ACTUALLY UPDATE CHANNEL TABLES.
35    PASS=1
      CALL UPDATE(NODE,DEST,CLASS,IDELAY,PASS)
      PASS=2
      CALL UPDATE(DEST,NODE,CLASS,IDELAY,PASS)
      IF(CLASS.EQ.2)GO TO 75
      GO TO 180
C CS ARRIVAL
30    NDEST=PARAM(1)/2
      KNODE=NODE-NDEST
      IF(IYESNO.EQ.1)GO TO 40
      IF(IYESNO.EQ.2)GO TO 40
      IYESNO=0
      IDELAY=0
      PASS=2
      CALL ROUTE(DEST,NODE,IYESNO,IDELAY,CLASS,PASS)
      IF(IYESNO.EQ.1)GO TO 40
      IF(IYESNO.EQ.2)GO TO 40
      CALLS(KNODE,1)=CALLS(KNODE,1)+1
      CALLS(KNODE,3)=CALLS(KNODE,3)+1
      GO TO 35
40    CALLS(KNODE,2)=CALLS(KNODE,2)+1
      GO TO 180
      END
      SUBROUTINE UPDATE(LNODE,LDEST,CLASS,IDELAY,PASS)
C
C*******************************************************************
C
C THIS ROUTINE UPDATES CHANTB ENTRIES FOR THE ROUTE SELECTED.
C
C*******************************************************************
C
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      NODE=LNODE
      DEST=LDEST
      INDEX=EVTBL(NODE,5)
```

```
      INODE=NODE
      RATIO=1
      IF(CLASS.EQ.2)RATIO=PARAM(4)
      FLAG=1
      IF(CLASS.EQ.1)KNODE=INODE-(PARAM(1)/2)
      IF(CLASS.EQ.2)INODE=DESTAB(NODE,DEST)
C DETERMINE IF DATA OR VOICE CONNECTION.
      IF(CLASS.EQ.1)GO TO 70
      IF(PASS.EQ.2)GO TO 150
10    IF(INODE.EQ.DEST)GO TO 50
      ICHAN=DESTAB(INODE,DEST)
      IF(ALTCH(ICHAN).EQ.1)GO TO 30
C NO ALT ROUTE
35    NLINES(ICHAN)=NLINES(ICHAN)-RATIO
      JCHNL=JARM3(ICHAN)-PARM3(ICHAN)+1
      ITOP=JCHNL+PARM3(ICHAN)-1
      I=JCHNL-1
      DO 15 J=1,RATIO
      JCHNL=I+1
C UPDATE CHANTB ENTRIES DEPENDING ON 1ST OR RETURN PATH.
      DO 20 I=JCHNL,ITOP
C MUST FIND FREE CHANNEL IN THE LINK.
      IF(CHANTB(I,4).EQ.0)GO TO 25
20    CONTINUE
      WRITE(6,204)
25    IF(FLAG.EQ.0)GO TO 40
      IF(CLASS.EQ.1)GO TO 26
      LINKTB(INODE,DEST)=ICHAN
      GO TO 27
80    IF(PASS.EQ.2)CHANTB(I,3)=CSARV(KDEST,2)+IDELAY
      IF(PASS.EQ.1)CHANTB(I,3)=CSARV(KNODE,2)+IDELAY
      IF(PASS.EQ.1)CHANTB(I,2)=EVTBL(NODE,1)+IDELAY
      IF(PASS.EQ.2)CHANTB(I,2)=EVTBL(DEST,1)+IDELAY
      GO TO 90
26    CSCHNL=JCHNL
27    FLAG=0
40    CHANTB(I,1)=DEST
      IF(CLASS.EQ.1)GO TO 80
      IF(PASS.EQ.1)CHANTB(I,2)=EVTBL(NODE,1)+IDELAY
      IF(PASS.EQ.2)CHANTB(I,2)=EVTBL(DEST,1)+IDELAY
      IF(PASS.EQ.1)CHANTB(I,3)=QUEUE(NODE,(INDEX+2))
      IF(PASS.EQ.2)CHANTB(I,3)=QUEUE(DEST,(QADDR+2))
C RATIO TELLS HOW MANY TIME SLOTS ARE REQUIRED.
90    CHANTB(I,4)=RATIO
      CHANTB(I,5)=NODCHL(ICHAN)
      CHANTB(I,7)=NODE
      CHANTB(I,8)=INDEX
      IF(CLASS.EQ.1)GO TO 16
15    CONTINUE
16    INODE=NODCHL(ICHAN)
C SHOULD WE KEEP LOOKING FOR A FREE CHANNEL?
      IF(DESTAB(INODE,DEST).EQ.0)GO TO 50
      GO TO 10
150   ITOP=PARAM(13)
      DO 160 INDEX=1,ITOP,6
C MUST FIND AN AVAILABLE QUEUE ENTRY.
      IF(QUEUE(NODE,INDEX).NE.100)GO TO 160
      IF(QUEUE(NODE,(INDEX+4)).NE.DEST)GO TO 160
      QADDR=EVTBL(DEST,5)
      IF(QUEUE(NODE,(INDEX+2)).EQ.QUEUE(DEST,(QADDR+2)))GO TO 10
160   CONTINUE
      WRITE(6,203)
      GO TO 50
30    ALTCH(ICHAN)=0
      ICHAN=DSTALT(INODE,DEST)
      GO TO 35
50    RETURN
70    DO 100 I=1,150,4
```

```
      IF(CALLQ(KNODE,I).EQ.O)GO TO 130
100   CONTINUE
130   IF(PASS.EQ.2)GO TO 140
C UPDATE BOTH CHANNELS FOR CIRCUIT SWITCH CONNECTION.
110   CALLQ(KNODE,I)=CSARV(KNODE,2)+IDELAY
      CALLQ(KNODE,(I+3))=LNODE
170   CALLQ(KNODE,(I+1))=LDEST
      CALLQ(KNODE,(I+2))=I
      INDEX=I
      GO TO 10
140   KDEST=DEST-(PARAM(1)/2)
      CALLQ(KNODE,I)=CSARV(KDEST,2)+IDELAY
      CALLQ(KNODE,(I+3))=LDEST
      GO TO 170
203   FORMAT('O','ERROR IN UPDATE 160')
204   FORMAT('O','ERROR IN UPDATE 20')
      END
      SUBROUTINE GETQ(LNODE,LDEST,CLASS,QADDR,CHPTR,PASS)
C
C*****************************************************************
C
C THIS ROUTINE IS USED WHEN IT IS NECESSARY TO TERMINATE
C A ROUTE.  IT FINDS THE STARTING CHANNEL ADDRESS
C FOR THE REMOVAL PROCESS.
C
C*****************************************************************
C
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
      FLAG=1
      NODE=LNODE
      DEST=LDEST
      INODE=NODE
      IF(CLASS.EQ.2)INODE=DESTAB(INODE,DEST)
      ICHAN=DESTAB(INODE,DEST)
20    IF(FLAG.EQ.O)ICHAN=DSTALT(INODE,DEST)
      JCHNL=JARM3(ICHAN)-PARM3(ICHAN)+1
      ITOP=JCHNL+PARM3(ICHAN)-1
      DO 10 J=JCHNL,ITOP
C TRY TO MATCH UP SOURCE, DEST, AND TIME.
      IF(CHANTB(J,7).NE.NODE)GO TO 10
      IF(CHANTB(J,1).NE.DEST)GO TO 10
      GO TO 40
50    QADDR=CHANTB(J,8)
      CHPTR=J
      GO TO 30
40    IF(PASS.EQ.2)GO TO 60
      IF(CHANTB(J,3).EQ.EVTBL(NODE,1))GO TO 50
      GO TO 10
60    IF(CHANTB(J,3).EQ.EVTBL(DEST,1))GO TO 50
10    CONTINUE
      IF(FLAG.EQ.O)GO TO 30
      FLAG=O
      GO TO 20
C FOUND MATCH-RETURN ADDRESS.
30    RETURN
C ERROR CONDITION-NO MATCH FOUND.
C70   WRITE(6,1000)
C     STIME=PARAM(9)
C     WRITE(6,1001)LNODE,LDEST,PASS,STIME
C     WRITE(6,1002)(EVTBL(NODE,K),K=1,5)
```

```
C     ITOP=EVTBL(LNODE,5)
C     ILIM=ITOP+5
C     WRITE(6,1004)(QUEUE(LNODE,M),M=ITOP,ILIM)
C     GO TO 30
1000  FORMAT('0','ERROR IN GETQ')
1001  FORMAT('0',4(I6,1X))
1002  FORMAT('0',5(I6,2X))
1004  FORMAT('0',6(I7,1X))
      END
C
      SUBROUTINE STATX
C
C*************************************************************
C
C THIS ROUTINE PRINTS OUT PERFORMANCE STATISTICS IN AN
C ABBREVIATED FORM.
C
C*************************************************************
C
      IMPLICIT INTEGER (A-S)
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
C
      WRITE(6,2000)
      WRITE(6,2011)
      WRITE(6,2012)
      WRITE(6,2001)(PARAM(I),I=1,16)
C CALCULATE THE THROUGHPUT AND UTILIZATION.
      WRITE(6,7002)
      ILIM=PARAM(2)
      NSITES=PARAM(1)/2
      PAKTOT=0
      UTOT=0.0
      DO 10 I=1,ILIM
      JCHNL=JARM3(I)-PARM3(I)+1
      ITOP=JCHNL+PARM3(I)-1
      UTIL1=0.0
      PSENT1=0
          DO 20 J=JCHNL,ITOP
          DUMMY=CHANTB(J,6)
          IF(CHANTB(J,4).EQ.0)GO TO 80
          DUMMY=DUMMY+PARAM(10)-CHANTB(J,2)
80        PSENT=CHANTB(J,10)
          PSENT1=PSENT1+PSENT
          UTIL=1.0*DUMMY/PARAM(10)
          UTIL1=UTIL1+UTIL
20        CONTINUE
      UTIL1=UTIL1/PARM3(I)
      NS=SORCHL(I)-NSITES
      ND=NODCHL(I)-NSITES
      WRITE(6,7003)I,PSENT1,UTIL1,NS,ND,PARM3(I)
      THRUTL(I,1)=PSENT1
      THRUTL(I,2)=UTIL1
      PAKTOT=PAKTOT+PSENT1
      UTOT=UTOT+UTIL1
10    CONTINUE
      PAKAVG=PAKTOT/ILIM
      UAVG=UTOT/ILIM
      XSECS=PARAM(10)/1000.
      PAKTHR=PAKAVG/XSECS + 0.5
      WRITE(6,7004)PAKAVG,UAVG,PAKTHR
C CALCULATE THE PACKET NODE STATISTICS.
```

```
      WRITE(6,7005)
      WRITE(6,7006)
      ITOP=PARAM(1)/2
      IF(PARAM(8).EQ.0)GO TO 31
      QTOT=0
      SUMPAK=0
      TOTDEL=0.0
      DO 30 I=1,ITOP
      QTOT=QTOT+QCNT(I)
      SUMPAK=SUMPAK+APCKTS(I)
      TOTDEL=TOTDEL+TDELAY(I)
      ZDELAY=TDELAY(I)/APCKTS(I)
      WRITE(6,7007)I,ZDELAY,QCNT(I)
      ZDBLK(I,1)=ZDELAY
30    CONTINUE
      ZQAVG=1.0*QTOT/ITOP
      ZDAVG=TOTDEL/SUMPAK
      GO TO 35
31    ZDELAY=0.0
      DO 32 I=1,ITOP
      WRITE(6,7007)I,ZDELAY,QCNT(I)
      ZDBLK(I,1)=ZDELAY
32    CONTINUE
      ZDAVG=0.0
      ZQAVG=0.0
35    CONTINUE
      WRITE(6,7008)ZDAVG,ZQAVG
C CALCULATE THE CS NODE STATISTICS
      WRITE(6,7009)
      WRITE(6,7010)
      BIGTOT=0
      ALOST=0
      AKEPT=0
      DO 40 I=1,ITOP
      K=ITOP+I
      ITOT=CALLS(I,1)+CALLS(I,2)
      ILOST=CALLS(I,2)
      IKEPT=CALLS(I,3)
      UTIL=0.0
      IF(ITOT.EQ.0)GO TO 50
      UTIL=1.0*CALLS(I,2)/ITOT
50    WRITE(6,7011)K,ITOT,ILOST,UTIL,IKEPT
      ZDBLK(I,2)=UTIL
      BIGTOT=BIGTOT+ITOT
      ALOST=ALOST+ILOST
      AKEPT=AKEPT+IKEPT
40    CONTINUE
      IF(BIGTOT.EQ.0)GO TO 71
      ZCALLS=1.0*BIGTOT/ITOP
      ZBLOCK=1.0*ALOST/BIGTOT
      ZSYS=1.0*AKEPT/ITOP
      GO TO 72
71    ZCALLS=0.0
      ZBLOCK=0.0
      ZSYS=0.0
72    WRITE(6,7012)ZCALLS,ZBLOCK,ZSYS
C WRITE OUT THE EVENT TYPE FREQUENCIES.
73    WRITE(6,7013)(EVENTX(I),I=1,4)
      RETURN
7001  FORMAT(1H1,10X,'CHECKPOINT: TIME=',I12,3X,'MS')
7002  FORMAT(1H0,5X,'CHAN',3X,'THROUGHPUT',3X,'UTILIZATION',
     1 1X,'SOURCE',3X,'DEST',3X,'SLOTS')
7003  FORMAT(1H ,5X,I3,3X,I10,4X,F8.3,3(4X,I4))
7004  FORMAT(1H0,9X,'AVG NO OF PACKETS PER LINK=',I10/10X,
     1 'AVG LINK UTILIZATION =',F6.3/10X,
     2 'AVG LINK THROUGHPUT (PACKETS/SEC)=',I10//)
7005  FORMAT(1H0,40X,'PACKET NODE SUMMARY'//)
7006  FORMAT(1H ,5X,'NODE',3X,'AVG PACKET DELAY (SEC)',3X,
```

```
     1 'DATA TRANSACTIONS IN SYSTEM'/)
7007  FORMAT(1H ,5X,I3,8X,F10.3,15X,I10)
7008  FORMAT(1HO,9X,'AVG PACKET DELAY (SEC)=',F8.3/10X,
     1 'AVG NO OF DATA TRANSACTIONS AT A NODE=',F8.1//)
7009  FORMAT(1HO,40X,'CS NODE SUMMARY'//)
7010  FORMAT(1H ,'NODE',5X,'TOTAL CALLS',5X,'CALLS LOST',9X,
     1 'BLOCKING',5X,'CALLS IN SYSTEM')
7011  FORMAT(1H ,2X,I2,11X,I5,10X,I5,12X,F5.3,10X,I5)
7012  FORMAT(1HO,9X,'AVG NO OF CALLS PER NODE=',F8.1/10X,
     1 'FRACTION OF CALLS BLOCKED=',F9.3/10X,
     2 'AVG NO OF CALLS IN SYSTEM PER NODE=',F8.1)
7013  FORMAT(1HO,19X,'CLASS 2 (DATA) ARRIVALS =',I10/20X,
     1 'CLASS 2 (DATA) DEPARTS  =',I10/20X,
     2 'CLASS 1 ( CS ) ARRIVALS =',I10/20X,
     3 'CLASS 1 ( CS ) DEPARTS  =',I10)
2000  FORMAT('1',10X,'SYSTEM PERFORMANCE MEASURES FOR THE ',
     1 'GIVEN INPUT PARAMETERS')
2001  FORMAT(' ',1X,3(I5,1X),I2,2X,I4,'MS ',I2,' MS',3X,I2,
     1'MIN',3X,I2,'SEC',2X,I5,' MS',I8,'MS ',I7,1X,
     2I5,'KBS ',I7,3X,I3,'SEC ',I4,'B',2X,I2)
2011  FORMAT('O',1X,'NODES LINKS SLOTS RATIO SLOT NODE  CS',6X,
     1'PS  MSG START TIME END TIME PACKET  VDR RATES ',
     2'Q SIZE   CS      PACKET PACKETS')
2012  FORMAT(' ',25X,'TIME DELAY ARRIVAL ARRIVAL',21X,
     1'LOADING',19X,'SERVICE SIZE   PER MSG')
8000  FORMAT(1HO)
8001  FORMAT(1HO,1X,'TIME  MEAN PAC AVG LINK  AVG NO PAC AVG ',
     1 'LINK  DATA     DATA    VOICE     VOICE    ',
     2 'MEAN Q  FRAC OF')
8002  FORMAT(2X,'INT   DELAY    UTILIZ.   PER LINK   ',
     1 'THRUPUT   ARRIVALS DEPARTS ARRIVALS DEPARTS ',
     2 'LENGTH  CALLS')
8003  FORMAT(2X,'(MIN) (SEC)      (%)      (DELTA)    ',
     1 '(PAC/SEC) PER INT  PER INT PER INT  PER INT ',
     2 '(TRAN)  BLOCKED'/)
8004  FORMAT(2X,I3,F10.3,F8.3,I12,I10,I10,I9,I8,I9,F7.1,F8.3)
      END
C*****************************************************************
C*
C*  SUBROUTINE OUTFAC ACCOMPLISHES THE TRANSFORMATION FROM THE
C*  SIMULATION DOMAIN BACK INTO THE PROBLEM DOMAIN.  IT ACTS AS
C*  AN INTERFACE (OR OUTERFACE) BETWEEN THE NETWORK TOPOLOGY
C*  PERFORMANCE GENERATOR (THE SIMULATOR) AND THE PERFORMANCE
C*  EVALUATION MODULE, PERFRM.
C*
C*****************************************************************
C
      SUBROUTINE OUTFAC
      IMPLICIT INTEGER (A-S)
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      LOGICAL A,B
      REAL PSWORK,CSWORK,CSSERV,CCOST,DIST,D,PFAIL,DSCONL,
     1     FRACL,DELAYL,BLOCKL
      REAL DELAY,BLOCK,AVGT,AVGU,AVGTPS,AVGD,AVGB
      COMMON/AREA2/EVTBL(52,5),DESTAB(52,52),DSTALT(52,52),
     1PARAM(17),CHANTB(8000,11),QUEUE(26,1800),CALLQ(26,200),
     2CUMTIM(26,13),CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     3QCNT(52),SORCHL(160),NODCHL(160),ALTCH(160),CSARV(26,3),
     4NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CUMLOD(26,26),
     5CUMCNT(26,26),ROUT(160),APCKTS(26),TDELAY(26),EVENTX(4),
     6THRUTL(160,2),ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,
     7PARM3(160),JARM3(160)
```

```
C
C WE NOW INITIALIZE THE PROBLEM DOMAIN PERFORMANCE VARIABLES FROM THE
C RESULTS OF THE SIMULATION.
      DO 10 I=1,NLINKS
      J=2*I-1
      NODETB(I,1)=SORCHL(J)-NSITES
      NODETB(I,2)=SORCHL(J+1)-NSITES
      THRU(I)=(THRUTL(J,1)+THRUTL(J+1,1))/2.0
      UTIL(I)=(THRUTL(J,2)+THRUTL(J+1,2))/2.0
10    CONTINUE
C
      DO 20 I=1,NSITES
      DELAY(I)=ZDBLK(I,1)
      BLOCK(I)=ZDBLK(I,2)
20    CONTINUE
C
      AVGT=PAKAVG
      AVGU=UAVG
      AVGTPS=PAKTHR
      AVGD=ZDAVG
      AVGB=ZBLOCK
      RETURN
      END
C
C*****************************************************************
C*
C*  SUBROUTINE PERFRM EXAMINES THE PERFORMANCE DATA AND IS THE
C*  DRIVER IN DETERMINING HOW THE TOPOLOGY WILL BE RECONFIGURED.
C*
C*****************************************************************
C
      SUBROUTINE PERFRM
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
      INTEGER FLAGD,FLAGB,FLAGU,FLAGR/0/,FEAS/0/,BCC(2,26),NBCC(2)
      IADD=0
C
C DETERMINE WHICH CONSTRAINTS ARE SATISFIED (FLAG=1).
      FLAGD=0
      FLAGB=0
      FLAGU=0
      IF(AVGD.LE.DELAYL)FLAGD=1
      IF(AVGB.LE.BLOCKL)FLAGB=1
      IF(AVGU.GE.UTILM)FLAGU=1
      IF(FLAGR.EQ.0)CALL BICON(FLAGR,BCC,NBCC)
      IF((FLAGD+FLAGB+FLAGR).EQ.3)FEAS=1
      WRITE(6,4001)FLAGD,FLAGB,FLAGU,FLAGR,FEAS
4001  FORMAT(1H0,5X,'FLAGD=',I2,7X,'FLAGB=',I2,7X,'FLAGU=',
     1 I2,7X,'FLAGR=',I2,7X,'FEAS=',I2)
      IF(FEAS.EQ.1)GO TO 400
      IF((FLAGD.EQ.1).AND.(FLAGB.EQ.1))GO TO 300
C
C EITHER DELAY OR BLOCKING (OR BOTH) IS NOT SATISFIED.  SEE IF WE
C CAN ADD CAPACITY TO EXISTING LINKS AND/OR ADD A LINK.
100   NADCAP=0
      LXFLAG=0
      DO 110 I=1,NLINKS
      IF(UTIL(I).LE.UTILM)GO TO 110
      J=NODETB(I,1)
      K=NODETB(I,2)
      DIFF=UTIL(I)-UTILM
```

```
      NTENS=INT(DIFF/.10)+1
      CONECT(J,K)=CONECT(J,K)+NTENS
      IF(CONECT(J,K).LE.NCAPS)GO TO 108
      LXFLAG=1
      CONECT(J,K)=NCAPS
      GO TO 109
108   NADCAP=NADCAP+1
109   CONECT(K,J)=CONECT(J,K)
110   CONTINUE
C
C IF ANY LINK IS ALREADY AT FULL CAPACITY AND STILL NEEDS
C ADDITIONAL CAPACITY, OR IF NO ADDITIONAL CAPACITY FOR ANY
C LINK IS INDICATED BUT THE B AND/OR D CONSTRAINTS ARE STILL
C NOT SATISFIED, THEN GO TO 300 AND ADD A LINK.
C OTHERWISE, RETURN.
      IF((NADCAP.EQ.0).OR.(LXFLAG.EQ.1))GO TO 300
      RETURN
300   CALL ADLINK(FLAGD,FLAGB,FLAGR,BCC,NBCC)
      RETURN
400   CALL DELINK(FLAGD,FLAGB,FLAGR,BCC,NBCC)
      RETURN
      END
C*****************************************************************
C*
C*  SUBROUTINE ADLINK DETERMINES WHERE A LINK SHOULD BE ADDED IN
C*  THE NETWORK AND THEN ADDS IT.  THE HEURISTIC USED IS A COMBINED
C*  SATURATED CUT-BICONNECTED COMPONENT METHOD.
C*
C*****************************************************************
      SUBROUTINE ADLINK(FLAGD,FLAGB,FLAGR,BCC,NBCC)
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
      INTEGER FLAGD,FLAGB,FLAGR,BCC(2,26),NBCC(2)
      INTEGER CUTSET(80),COMP1(26),COMP2(26),COMPOF(26)
      DIMENSION DUMMY(80),STDEVS(26,2)
C
      IF((FLAGD+FLAGB).LT.2)GO TO 300
C
C IN THIS CASE, SINCE DELAY AND BLOCKING ARE ALREADY SATISFIED,
C WE MAKE A CONNECTION BETWEEN THE TWO BCC'S WHICH GIVES US THE
C LEAST COST LINK.  FLAGR=0 WHEN THIS SECTION IS EXECUTED.
      KK=0
      LL=0
      SMDIST=999999.
      MAX1=NBCC(1)
      MAX2=NBCC(2)
      DO 220 I=1,MAX1
         DO 210 J=1,MAX2
         IF(DIST(BCC(1,I),BCC(2,J)).GE.SMDIST)GO TO 210
         KK=BCC(1,I)
         LL=BCC(2,J)
         SMDIST=DIST(BCC(1,I),BCC(2,J))
210      CONTINUE
220   CONTINUE
      CONECT(KK,LL)=1
      CONECT(LL,KK)=1
      IADD=1
      WRITE(6,8066)KK,LL
      RETURN
C
C FIND THE SATURATED CUT (CUTSET) BASED ON LINK UTILIZATION.
```

```
300    DO 305 I=1,NLINKS
       CUTSET(I)=0
       DUMMY(I)=UTIL(I)
305    CONTINUE
C FIRST, INITIALIZE THE LOGICAL MATRIX A TO THE CONECT MATRIX.
       NM1=NSITES-1
       DO 40 I=1,NM1
       A(I,I)=.FALSE.
       L=I+1
           DO 39 J=L,NSITES
           IF(CONECT(I,J).EQ.0)GO TO 38
           A(I,J)=.TRUE.
           A(J,I)=.TRUE.
           GO TO 39
38         A(I,J)=.FALSE.
           A(J,I)=.FALSE.
39         CONTINUE
40     CONTINUE
       A(NSITES,NSITES)=.FALSE.
C
C FIND THE MOST UTILIZED LINK AND PLACE IT IN THE CUTSET.
307    BIG=DUMMY(1)
       K=1
       DO 310 I=2,NLINKS
       IF(DUMMY(I).LE.BIG)GO TO 310
       K=I
       BIG=DUMMY(I)
310    CONTINUE
       DUMMY(K)=0.0
       CUTSET(K)=1
C
C REMOVE THAT LINK AND SEE IF THE NETWORK IS STILL CONNECTED.  IF
C IT IS STILL CONNECTED THEN GO TO 307 AND REPEAT UNTIL WE HAVE
C REMOVED ENOUGH LINKS SO THAT THE NETWORK IS DISCONNECTED.  THE
C LINKS REMOVED FORM THE CUTSET.
       L=NODETB(K,1)
       M=NODETB(K,2)
       A(L,M)=.FALSE.
       A(M,L)=.FALSE.
       IDISC=0
       CALL CLOZUR(IDISC)
       IF(IDISC.EQ.0)GO TO 307
C
C THE NETWORK IS NOW DISCONNECTED, CONSISTING OF 2 CONNECTED
C COMPONENTS, COMP1 AND COMP2.  FIND COMP1 AND COMP2 FROM B.
       IROW=0
318    IROW=IROW+1
       NCOMP1=0
       NCOMP2=0
       DO 320 I=1,NSITES
       IF(B(IROW,I))GO TO 319
       NCOMP1=NCOMP1+1
       COMP1(NCOMP1)=I
       COMPOF(I)=1
       GO TO 320
319    NCOMP2=NCOMP2+1
       COMP2(NCOMP2)=I
       COMPOF(I)=2
320    CONTINUE
       IF((NCOMP1.EQ.0).OR.(NCOMP2.EQ.0))GO TO 318
       WRITE(6,3005)(COMP1(I),I=1,NCOMP1)
       WRITE(6,3006)(COMP2(I),I=1,NCOMP2)
3005   FORMAT(1H0,2X,'COMP1=',30I3)
3006   FORMAT(3X,'COMP2=',30I3)
C
C CALCULATE THE STDEVS TO BE USED BY THE HEURISTIC.
       SUMD=0.0
       SUMB=0.0
```

```
      SSD=0.0
      SSB=0.0
      DO 350 I=1,NSITES
      SUMD=SUMD+DELAY(I)
      SUMB=SUMB+BLOCK(I)
      SSD=SSD+DELAY(I)*DELAY(I)
      SSB=SSB+BLOCK(I)*BLOCK(I)
350   CONTINUE
      VARD=(SSD-(SUMD*SUMD)/NSITES)/(NSITES-1)
      VARB=(SSB-(SUMB*SUMB)/NSITES)/(NSITES-1)
      SD=SQRT(VARD)
      SB=SQRT(VARB)
C
C AT THIS POINT, AT LEAST ONE OF THE TWO FLAGS IS O.
      IF((FLAGD.EQ.O).AND.(FLAGB.EQ.O))GO TO 380
      IF(FLAGB.EQ.O)GO TO 370
C
C ONLY FLAGD=O SO USE ONLY DELAY STATS.
360   DO 365 I=1,NSITES
      STDEVS(I,1)=(DELAY(I)-AVGD)/SD
      STDEVS(I,2)=0.0
365   CONTINUE
      GO TO 388
C
C ONLY FLAGB=O SO USE ONLY BLOCKING STATS.
370   DO 375 I=1,NSITES
      STDEVS(I,1)=0.0
      STDEVS(I,2)=(BLOCK(I)-AVGB)/SB
375   CONTINUE
      GO TO 388
C
C BOTH ARE O, SO USE BOTH BLOCKING AND DELAY STATS.
380   DO 385 I=1,NSITES
      STDEVS(I,1)=(DELAY(I)-AVGD)/SD
      STDEVS(I,2)=(BLOCK(I)-AVGB)/SB
385   CONTINUE
388   IF(FLAGR.EQ.1)GO TO 390
C
C IN THIS CASE, MAKE A CONNECTION BETWEEN THE TWO BCC'S THAT
C ALSO SPANS THE CUT, IF POSSIBLE.  IF IT IS NOT POSSIBLE,
C MAKE 2 CONNECTIONS:  ONE FOR THE CUT AND ONE FOR THE
C BICONNECTIVITY (I.E., RELIABILITY).
      KK=O
      LL=O
      BIG=-99.
      MAX1=NBCC(1)
      MAX2=NBCC(2)
      DO 340 I=1,MAX1
      K=BCC(1,I)
          DO 335 J=1,MAX2
          L=BCC(2,J)
          KCOMP=COMPOF(K)
          LCOMP=COMPOF(L)
          IF(KCOMP.EQ.LCOMP)GO TO 335
          COMB=STDEVS(K,1)+STDEVS(L,1)+STDEVS(K,2)+STDEVS(L,2)
          IF(COMB.LE.BIG)GO TO 335
          BIG=COMB
          KK=K
          LL=L
335       CONTINUE
340   CONTINUE
      IF(KK.EQ.O)GO TO 341
      CONECT(KK,LL)=3
      CONECT(LL,KK)=3
      IADD=1
      WRITE(6,8066)KK,LL
      RETURN
C
```

```
C UNFORTUNATELY, IT IS NOT POSSIBLE TO USE THE SAME LINK TO
C SPAN BOTH THE BCC'S AND THE SATURATED CUT.  THUS, WE ADD TWO
C NEW LINKS - ONE FROM BCC(1) TO BCC(2) AND THE OTHER ACROSS
C THE CUT.  THE LINK DETERMINED HERE SPANS THE BCC'S.
341   BIG=-99.
      KK=0
      LL=0
      DO 343 I=1,MAX1
      K=BCC(1,I)
          DO 342 J=1,MAX2
          L=BCC(2,J)
          COMB=STDEVS(K,1)+STDEVS(L,1)+STDEVS(K,2)+STDEVS(L,2)
          IF(COMB.LE.BIG)GO TO 342
          BIG=COMB
          KK=K
          LL=L
342       CONTINUE
343   CONTINUE
      CONECT(KK,LL)=1
      CONECT(LL,KK)=1
      IADD=1
      WRITE(6,8066)KK,LL
C
C THE MAX COMB (AT NODES KK,LL) VALUE WILL DETERMINE THE LINK
C TO BE ADDED.  THE LINK DETERMINED HERE SPANS THE SATURATED CUT.
390   BIG=-99.
      KK=0
      LL=0
      DO 395 I=1,NCOMP1
      K=COMP1(I)
          DO 394 J=1,NCOMP2
          L=COMP2(J)
          IF(CONECT(K,L).NE.0)GO TO 394
          COMB=STDEVS(K,1)+STDEVS(L,1)+STDEVS(K,2)+STDEVS(L,2)
          IF(COMB.LE.BIG)GO TO 394
          BIG=COMB
          KK=K
          LL=L
394       CONTINUE
395   CONTINUE
      CONECT(KK,LL)=3
      CONECT(LL,KK)=3
      IADD=1
      WRITE(6,8066)KK,LL
8066  FORMAT(1H0,5X,'A NEW LINK WILL BE ADDED BETWEEN NODES',I5,
     1 '    AND',I5)
      RETURN
      END
C
C*****************************************************************
C*
C*  SUBROUTINE DELINK CONTROLS THE DECREASE IN CAPACITY OF THE
C*  NETWORK AS WELL AS ANY LINK DELETIONS IN THE NETWORK.  IT
C*  ALSO SAVES THE CURRENTLY BEST TOPOLOGY WHICH IS RESTORED IN
C*  CASE THE PERTURBATION RESULTS IN AN UNSATISFIED CONSTRAINT.
C*  A CONSECUTIVE NUMBER (LIMIT) OF MINOR PERTURBATIONS WHICH
C*  FAIL TO SATISFY THE CONSTRAINTS INDICATES THAT A LOCAL MIN-
C*  IMUM HAS BEEN REACHED.
C*
C*****************************************************************
      SUBROUTINE DELINK(FLAGD,FLAGB,FLAGR,BCC,NBCC)
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
```

```
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B,FAILED(26,26)/676*.FALSE./
      INTEGER FLAGD,FLAGB,FLAGR,BCC(2,26),NBCC(2)
      INTEGER BESTC(26,26),BODETB(80,2),BLINKS
      DIMENSION DUMMY(80)
      DATA L,M,IDIFF/0,0,0/
C
C TO REACH DELINK, WE MUST HAVE HAD AT SOME TIME FLAGD=FLAGB=
C FLAGR=1 (WHICH SETS FEAS=1).  WE NOW TRY TO OBTAIN THE BEST
C FEASIBLE SOLUTION POSSIBLE.  WE ARE REALLY MAXIMIZING THE LINK
C UTILIZATION AT THIS POINT.
      LIMIT=4
C
C IF THIS PERTURBATION RESULTED IN THE DELAY(D) OR BLOCKING(B)
C CONSTRAINT NOT BEING SATISFIED, THEN BUMP THE # OF FAILURES BY
C 1 AND REVERT BACK TO THE PREVIOUS BEST TOPOLOGY.  IF D AND B
C ARE STILL SATISFIED, STORE THIS TOPOLOGY AS THE BEST TOPOLOGY
C AND CONTINUE BY PERTURBING THIS NEW TOPOLOGY.
      IF((FLAGD+FLAGB).EQ.2)GO TO 600
      IFAIL=IFAIL+1
      WRITE(6,8070)IFAIL
8070  FORMAT(1HO,'IFAIL=',I3,'   THE PERTURBED TOPOLOGY ',
     1 'FAILED TO SATISFY ALL THE CONSTRAINTS.  REVERT BACK TO ',
     2 'BEST TOPOLOGY.')
      IF(IFAIL.EQ.LIMIT)GO TO 800
      IF(IDIFF.GT.1)GO TO 790
      IF(IRSAVE.EQ.1)IRSAVE=2
      DO 505 I=1,NSITES
      DO 505 J=1,NSITES
      CONECT(I,J)=BESTC(I,J)
505   CONTINUE
      GO TO 700
C
C STORE THE BEST TOPOLOGY.
600   WRITE(6,9000)
9000  FORMAT(1HO,5X,'A FEASIBLE SOLUTION HAS BEEN FOUND.'/6X,
     1 'STORE IT AND MAKE PERTURBATIONS FROM THIS TOPOLOGY.')
      IFAIL=0
      BLINKS=NLINKS
      BESTU=AVGU
      BESTD=AVGD
      BESTB=AVGB
      BCOST=COST
      DO 605 I=1,NSITES
      DO 605 J=1,NSITES
      BESTC(I,J)=CONECT(I,J)
605   CONTINUE
      DO 610 I=1,NLINKS
      DUMMY(I)=UTIL(I)
      BODETB(I,1)=NODETB(I,1)
      BODETB(I,2)=NODETB(I,2)
610   CONTINUE
C
C REDUCE THE NETWORK CAPACITY OF THE CURRENTLY BEST TOPOLOGY.
C FIRST FIND THE LEAST UTILIZED LINK.
700   SMALL=1.0
      K=0
      DO 705 I=1,BLINKS
      J1=BODETB(I,1)
      J2=BODETB(I,2)
      IF(FAILED(J1,J2))GO TO 705
      IF(DUMMY(I).GE.SMALL)GO TO 705
      K=I
      SMALL=DUMMY(I)
705   CONTINUE
      IF(K.EQ.0)GO TO 795
      L=BODETB(K,1)
      M=BODETB(K,2)
```

```
      DUMMY(K)=1.0
      IF(CONECT(L,M).GT.1)GO TO 750
      IDIFF=CONECT(L,M)
      CONECT(L,M)=0
      CONECT(M,L)=0
      WRITE(6,8087)L,M
8087  FORMAT(1H0,5X,'CHECK TO SEE IF LINK (',I2,',',I2,
     1 ') CAN BE DELETED WITHOUT DESTROYING THE BICONNECTIVITY.')
      FLAGR=0
      CALL BICON(FLAGR,BCC,NBCC)
C
C IF DELETING LINK (L,M) CAUSED THE BICONNECTIVITY TO DISSOLVE,
C THEN WE CANNOT DELETE (L,M). MARK IT FAILED SO WE DON'T EVEN
C CHECK IT THE NEXT TIME.
      IF(FLAGR.EQ.1)GO TO 710
      FAILED(L,M)=.TRUE.
      FAILED(M,L)=.TRUE.
      FLAGR=1
      CONECT(L,M)=IDIFF
      CONECT(M,L)=IDIFF
      GO TO 700
C
C DELETING LINK (L,M) DOES NOT DESTROY THE BICONNECTIVITY, SO GO
C SIMULATE AFTER DELETING THIS LINK.
710   IADD=-1
      WRITE(6,8088)K,L,M,IDIFF
8088  FORMAT(1H0,5X,'LINK #',I3,'  WHICH CONNECTS NODES',I3,
     1 '  AND',I3,',AND WHICH IS OF CAPACITY TYPE',I2,
     2 '  IS DELETED.')
      RETURN
C
C DECREASE THE CAPACITY OF LINK (L,M) ON THE BASIS OF HOW MUCH
C LESS IT IS THAN UTILM.
750   IOLD=CONECT(L,M)
      DIFF=UTILM-SMALL
      NTENS=INT(DIFF/.10)+1
      IF(UTILM.LT.SMALL)NTENS=1
      CONECT(L,M)=CONECT(L,M)-NTENS
      IF(CONECT(L,M).LE.0)CONECT(L,M)=1
      CONECT(M,L)=CONECT(L,M)
      IDIFF=IOLD-CONECT(L,M)
760   WRITE(6,8089)K,L,M,IDIFF
8089  FORMAT(1H0,5X,'LINK #',I3,'  WHICH CONNECTS NODES',I3,
     1 '  AND',I3,'  IS REDUCED IN CAPACITY BY',I2,'  UNITS.')
      RETURN
C
C WE REDUCED THE CAPACITY BY TOO MUCH, SO DECREASE THE REDUCTION
C AND REPEAT.
790   CONECT(L,M)=CONECT(L,M)+IDIFF-1
      IDIFF=1
      CONECT(M,L)=CONECT(L,M)
      GO TO 760
C
C WE HAVE EXHAUSTED ALL POSSIBLE PERTURBATIONS FROM THIS TOPOLOGY,
C ALL WITHOUT SUCCESS, SO WE TERMINATE AT THIS LOCAL MINIMUM.
795   WRITE(6,8094)
8094  FORMAT(1H0,5X,'ALL PERTURBATIONS HAVE BEEN UNSUCCESSFUL.  ',
     1 'THE LOCAL MINIMUM TOPOLOGY IS AS FOLLOWS:')
      GO TO 805
800   WRITE(6,8090)
8090  FORMAT(1H0,5X,'THE CONSECUTIVE FAILURE LIMIT HAS BEEN ',
     1 'REACHED, INDICATING THAT WE HAVE FOUND A LOCAL MINIMUM.'/6X,
     2 'THE FOLLOWING TOPOLOGICAL DATA REPRESENTS THE LOCAL MIN:')
805   WRITE(6,8091)(I,I=1,NSITES)
8091  FORMAT(1H0,5X,'THE BEST CONECT MATRIX:'/6X,30I2)
      DO 801 I=1,NSITES
      WRITE(6,8092)I,(BESTC(I,J),J=1,NSITES)
8092  FORMAT(4X,31I2)
```

```
801    CONTINUE
       WRITE(6,8093)BLINKS,BESTU,BESTD,BESTB,BCOST
8093   FORMAT(1HO,5X,'BLINKS=',I4/6X,'BESTU=',F6.3/6X,
      1 'BESTD=',F6.3/6X,'BESTB=',F6.3/6X,'COST OF THIS ',
      2 'TOPOLOGY=',F10.2)
       WRITE(6,8095)(I,I=1,NSITES)
8095   FORMAT(1HO,5X,'FAILED MATRIX:'/6X,30I2)
       DO 810 I=1,NSITES
       WRITE(6,8096)I,(FAILED(I,J),J=1,NSITES)
810    CONTINUE
8096   FORMAT(4X,I2,30L2)
820    CONECT(L,M)=CONECT(L,M)+IDIFF
       CONECT(M,L)=CONECT(L,M)
       CALL RELY
       STOP
       END
C
C*****************************************************************
C*
C*  SUBROUTINE CLOZUR COMPUTES THE TRANSITIVE CLOSURE OF A MATRIX
C*  (GRAPH) AND DETERMINES IF THE GRAPH IS DISCONNECTED (IDISC=1).
C*
C*****************************************************************
C
       SUBROUTINE CLOZUR(IDISC)
       COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
      1        CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
      2        PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
      3        D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
      4        DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
      5        NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
      6        BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
       INTEGER CONECT,HEAD,DEG,PRED,SUCC
       LOGICAL A,B
C COPY A TO WORK AREA B.
42     DO 43 I=1,NSITES
       DO 43 J=1,NSITES
       B(I,J)=A(I,J)
43     CONTINUE
C COMPUTE THE TRANSITIVE CLOSURE OF THE GRAPH(NETWORK).
       DO 50 I=1,NSITES
       DO 50 J=1,NSITES
       IF(.NOT.B(J,I))GO TO 50
            DO 45 K=1,NSITES
            B(J,K)=B(J,K).OR.B(I,K)
45          CONTINUE
50     CONTINUE
C
C CHECK IF B IS DISCONNECTED.
       DO 60 I=1,NM1
       L=I+1
            DO 55 J=L,NSITES
            IF(B(I,J))GO TO 55
            IDISC=1
            GO TO 77
55          CONTINUE
60     CONTINUE
77     RETURN
       END
C*************************************************************
C*
C*  SUBROUTINE BICON CHECKS TO SEE IF THE GRAPH IS (NODE)
C*  BICONNECTED.  IF IT IS NOT, BICON OUTPUTS THE FIRST TWO
C*  BICONNECTED COMPONENTS, BCC(1) AND BCC(2), HAVING MORE
C*  THAN 2 NODES IN THEM; AND FROM THIS, THE NEW LINK TO BE
C*  ADDED IS DETERMINED.
C*
C*************************************************************
```

```
C
      SUBROUTINE BICON(FLAGR,BCC,NBCC)
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1        CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2        PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3        D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4        DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5        NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6        BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B,BC(26)
      INTEGER FLAGR,BCC(2,26),NBCC(2),LOW(26),DF(26),NEW(26)
      INTEGER STACK(80,2)/160*0/,FATHER(26),V,W,COUNT,TOP,NEXT(26)
C
C INITIALIZE ALL DATA STRUCTURES
      COUNT=1
      TOP=0
      NTWOS=0
      DO 10 I=1,NSITES
      LOW(I)=0
      DF(I)=0
      NEW(I)=1
      FATHER(I)=0
      BCC(1,I)=0
      BCC(2,I)=0
      NEXT(I)=1
10    CONTINUE
C
C BEGIN DEPTH FIRST SEARCH FOR THE BCC'S.
      N=1
      V=1
11    NEW(V)=0
      DF(V)=COUNT
      COUNT=COUNT+1
      LOW(V)=DF(V)
15    J=NEXT(V)
      IF(J.GT.NSITES)GO TO 30
      DO 20 W=J,NSITES
      NEXT(V)=NEXT(V)+1
      IF(CONECT(V,W).EQ.0)GO TO 20
C
C PUT EDGE (V,W) ON THE STACK IF IT IS NOT ALREADY THERE.
      IF((DF(V).LT.DF(W)).AND.(NEW(W).EQ.0))GO TO 19
      IF((DF(V).GT.DF(W)).AND.(W.EQ.FATHER(V)))GO TO 20
      TOP=TOP+1
      STACK(TOP,1)=V
      STACK(TOP,2)=W
C     WRITE(6,60)TOP,V,W
C60   FORMAT(1H ,'TOP=',I3,'  V=',I2,'  W=',I2)
      IF(NEW(W).EQ.0)GO TO 19
      FATHER(W)=V
      V=W
      GO TO 11
19    IF(W.EQ.FATHER(V))GO TO 20
      LOW(V)=MIN0(LOW(V),DF(W))
20    CONTINUE
C
C WE HAVE EXHAUSTED THE ADJACENCY LIST OF V.
30    W=V
      V=FATHER(W)
      IF(V.EQ.0)GO TO 200
      IF(LOW(W).LT.DF(V))GO TO 90
C
C WE HAVE FOUND A BCC.  POP OFF THE STACK THOSE NODES THAT
C BELONG TO THIS BCC.
      IDONE=0
      DO 70 I=1,NSITES
      BC(I)=.FALSE.
```

```
70     CONTINUE
85     N1=STACK(TOP,1)
       N2=STACK(TOP,2)
C      WRITE(6,62)N1,N2,V,W
C62    FORMAT(1H ,'N1=',I3,'    N2=',I3,'   V=',I3,'   W=',I3)
       STACK(TOP,1)=0
       STACK(TOP,2)=0
       TOP=TOP-1
       IF((N1.EQ.V).AND.(N2.EQ.W))IDONE=1
       BC(N1)=.TRUE.
       BC(N2)=.TRUE.
       IF(IDONE.EQ.1)GO TO 86
       IF(TOP.EQ.0)GO TO 86
       GO TO 85
86     L=0
       DO 71 I=1,NSITES
       IF(.NOT.BC(I))GO TO 71
       IF(I.EQ.V)GO TO 71
       L=L+1
       BCC(N,L)=I
71     CONTINUE
       NBCC(N)=L
       WRITE(6,3000)N,(BCC(N,I),I=1,L)
3000   FORMAT(1H0,5X,'BCC # =',I2,'   CONTAINS THE FOLLOWING ',
     1 'NODES:'/5X,30I3)
       WRITE(6,3001)V
3001   FORMAT(6X,'WHILE THE ARTICULATION POINT',I3,
     1 '  IS NOT INCLUDED IN THE BCC.')
       IF(L.GE.NM1)GO TO 99
       IF(L.EQ.1)GO TO 89
       IF(N.EQ.2)GO TO 100
       N=2
       GO TO 90
89     IVART=V
       NTWOS=NTWOS+1
       IF(NTWOS.GE.NM1)GO TO 200
90     LOW(V)=MIN0(LOW(V),LOW(W))
       GO TO 15
99     FLAGR=1
       GO TO 200
100    IF(NTWOS.EQ.0)GO TO 200
       DO 101 I=1,L
       IF(BCC(2,I).NE.IVART)GO TO 101
       IF(I.EQ.L)GO TO 103
       LM1=L-1
           DO 102 J=I,LM1
           BCC(2,J)=BCC(2,J+1)
102        CONTINUE
       GO TO 103
101    CONTINUE
       GO TO 200
103    IF((NBCC(1)+NTWOS+L).GE.NM1)GO TO 104
       BCC(2,L)=0
       L=LM1
       NBCC(2)=L
       GO TO 105
104    BCC(2,L)=V
105    WRITE(6,3002)(BCC(N,I),I=1,L)
3002   FORMAT(1H0,5X,'BCC # 2 IS RESTRUCTURED AS FOLLOWS:'/5X,
     1 30I3)
200    RETURN
       END
C
C *************************************************************
C *
C *  SUBROUTINE RELY CALCULATES RELIABILITY MEASURES (I.E.,
C *  PROBABILITY OF A DISCONNECT, FRACTION OF NODES UNABLE TO
C *  COMMUNICATE) FOR THE LOCAL MINIMUM OBTAINED.
```

```
C *
C ***************************************************************
C
      SUBROUTINE RELY
      COMMON/AREA1/X(26),Y(26),PSWORK(26,26),CSWORK(26,26),CSSERV(26),
     1       CCOST(5,3),DIST(26,26),CONECT(26,26),HEAD(26),DEG(26),
     2       PRED(26),SUCC(26),A(26,26),B(26,26),INODE(26,26),MAP(26),
     3       D(26,26),NCAPS,NPACMS,NBITPK,KCON,PFAIL,DSCONL,FRACL,
     4       DELAYL,THRUM,UTILM,BLOCKL,ISKIP,ISEED,NSITES,NM1,NLINKS,
     5       NEEDTB(52,4),NODETB(80,2),THRU(80),UTIL(80),DELAY(26),
     6       BLOCK(26),AVGT,AVGU,AVGTPS,AVGD,AVGB,IADD,IRSAVE,COST
      INTEGER CONECT,HEAD,DEG,PRED,SUCC
      LOGICAL A,B
      INTEGER DISCON
      REAL RN
C
C MAXIMUM NO. OF POSSIBLE COMMUNICATING PAIRS OF NODES.
      MAX=NSITES*(NSITES-1)/2
C CONVERT THE CONNECTIVITY MATRIX TO THE BOOLEAN MATRIX A,
C AND DETERMINE HOW MANY LINKS ARE IN THE GRAPH.
      NLINKS=0
      DO 10 I=1,NM1
      A(I,I)=.FALSE.
      L=I+1
          DO 5 J=L,NSITES
          IF(CONECT(I,J).EQ.0)GO TO 3
          A(I,J)=.TRUE.
          A(J,I)=.TRUE.
          NLINKS=NLINKS+1
          GO TO 5
3         A(I,J)=.FALSE.
          A(J,I)=.FALSE.
5         CONTINUE
10    CONTINUE
      A(NSITES,NSITES)=.FALSE.
C
C DETERMINE HOW MANY SIMULATIONS ARE NEEDED FOR A 90% CONFIDENCE
C INTERVAL (1.645) SUCH THAT THE TRUE DISCONNECT PROBABILITY IS
C WITHIN +/- .02 OF THE OBSERVED PROBABILITY OF DISCONNECT.
C N=(Z**2)(P(1-P))/(D**2).
C FIRST, DETERMINE AN UPPER BOUND FOR P(THE TRUE PROBABILITY OF
C A DISCONNECT).
      UBOUND=1.0-((1.0-PFAIL)**NLINKS)
      NSIMS=(1.645**2)*UBOUND*(1.0-UBOUND)/(.02**2)
      NSIMS=NSIMS+1
C WRITE HEADER INFORMATION
      WRITE(6,2000)
2000  FORMAT(1H0,3X,'NO. OF    PROBABILITY    FRACTION OF NODES',
     1 '  PROBABILITY  OF  A'/5X,'SIMS   OF LINK FAILURE',
     2 '  UNABLE TO COMMUN.  NETWORK DISCONNECT    NLINKS')
C
      NUMDIS=0
      FRAC=0.0
C
C DO LOOP 80 ONCE FOR EACH SIMULATION.
C
          DO 80 NR=1,NSIMS
C COPY A TO B WHERE B IS A WORK AREA IN WHICH LINKS ARE
C PROBABILISTICALLY DELETED.
          DO 20 I=1,NSITES
          DO 20 J=1,NSITES
          B(I,J)=A(I,J)
20        CONTINUE
C
C DESTROY LINKS AT RANDOM.
          DO 30 I=1,NM1
          L=I+1
              DO 25 J=L,NSITES
```

```
            IF(.NOT.B(I,J))GO TO 25
            CALL RANDUM(RN,SEED)
            IF(RN.GE.PFAIL)GO TO 25
            B(I,J)=.FALSE.
            B(J,I)=.FALSE.
25          CONTINUE
30        CONTINUE
C
C COMPUTE THE TRANSITIVE CLOSURE (CONNECTIVITY) OF THE GRAPH.
          DO 40 I=1,NSITES
          DO 40 J=1,NSITES
          IF(.NOT.B(J,I))GO TO 40
            DO 35 K=1,NSITES
            B(J,K)=B(J,K).OR.B(I,K)
35          CONTINUE
40        CONTINUE
C
C CHECK TO SEE HOW MANY NODE PAIRS ARE STILL CONNECTED.
          DISCON=0
          DO 50 I=1,NM1
          L=I+1
            DO 45 J=L,NSITES
            IF(B(I,J))GO TO 45
            DISCON=DISCON+1
45          CONTINUE
50        CONTINUE
          FRAC=FRAC+1.0*DISCON/MAX
          IF(DISCON.GT.0)NUMDIS=NUMDIS+1
80        CONTINUE
C
C PRINT OUT FRACTION OF NODES UNABLE TO COMMUNICATE AND PROBABILITY
C OF A NETWORK DISCONNECT FOR THIS PROBABILITY OF FAILURE LEVEL.
      FRAC=FRAC/NSIMS
      DISC=1.0*NUMDIS/NSIMS
      WRITE(6,2001)NSIMS,PFAIL,FRAC,DISC,NLINKS
2001  FORMAT(1H ,2X,I5,6X,F9.3,10X,F8.4,11X,F8.4,5X,I10)
      RETURN
      END
```

APPENDIX B

Appendix B provides additional documentation for the program. Variable names are listed, and a description of each of these variables is given. The arguments given for the arrays reflect the dimensions required.

VARIABLE	Description
NSITES	Number of sites or locations. It is assumed that one packet switch (PS) node and one circuit switch (CS) node are located at each site.
X(NSITES)	X-coordinate for each site.
Y(NSITES)	Y-coordinate for each site.
PSWORK(NSITES, NSITES)	PS (data) traffic load between node pairs.
CSWORK(NSITES, NSITES)	CS (voice) traffic load between node pairs.
CSSERV(NSITES)	Average voice service time for calls originating at a given node.
NCAPS	Number of capacities or line types available.
CCOST(NCAPS, 3)	Capacity and cost matrix. (i, 1) = capacity (BPS) for line type i. (i, 2) = cost ($) per unit length of line type i. (i, 3) = fixed cost ($) of line type i.
DIST(NSITES, NSITES)	Distance matrix that stores the distance between each node pair.
CONECT(NSITES, NSITES)	Connectivity matrix. Entries can take on the values 0, 1, 2,..., NCAPS. It tells which nodes are connected and with which line type. A "0" indicates no connection.
HEAD(NSITES)	Pointers to the heads of linked lists. Used in generating starting topologies.
DEG(NSITES)	Degree of each node. Used in generating starting topologies.

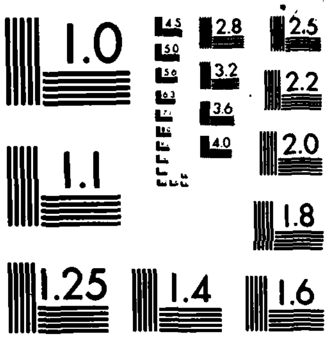
VARIABLE	Description
PRED(NSITES)	Predecessor of a node in a linked list.
SUCC(NSITES)	Successor of a node in a linked list.
A(NSITES, NSITES)	Logical equivalent of CONECT.
B(NSITES, NSITES)	Logical work area in which transitive closures of matrix A are computed.
INODE(NSITES, NSITES)	Matrix storing the first intermediate node on the shortest path between every pair of nodes. It is created in FLOYDS and used in TOPGEN for assigning capacities. It is also used in INFACE to construct the routing tables.
MAP(NSITES)	Stores the original node number for each of the newly created randomized nodes. A mapping from the randomized nodes onto the original nodes.
D(NSITES, NSITES)	Used in FLOYDS to store the distance of the shortest path between every node pair; also used in TOPGEN to store the estimated loading (BPS) on each link in the network.
NPACMS	Average number of packets per message.
NBITPK	Number of bits per packet.
KCON	Node connectivity constraint.
PFAIL	Probability of link failure.
DISCONL	Network disconnect probability limit.
FRACL	Limit on the fraction of nodes unable to communicate.

VARIABLE	Description
DELAYL	Mean packet delay constraint.
THRUM	Minimum desired throughput in average number of packets per second.
UTILM	Minimum average link utilization.
BLOCKL	Voice call blocking constraint.
ISKIP	Indicates whether a starting topology is to be generated (0) or a starting topology is provided (1). Also indicates if performance statistics are already available for a given topology (2).
ISEED	User provided seed to start the topology generation process.
NM1	NSITES-1. Used for loop control.
NLINKS	Total number of full-duplex links in a topology.
NEEDTB(2*NSITES, 4)	Storage space for the user provided seeds that are used by the simulator. Used to initialize the simulator seed tables prior to starting each simulation run.
NODETB(MAXL, 2)	Stores the two nodes that are the end points for a given link. MAXL denotes the maximum number of links allowed. Currently, MAXL = 80.
THRU(MAXL)	Stores the throughput (number of packets) for each link in the network.
UTIL(MAXL)	Stores the link utilization for each link in the network.

VARIABLE	Description
DELAY(NSITES)	Stores the mean packet delay at each PS node in the network.
BLOCK(NSITES)	Stores the fraction of voice calls blocked at each CS node in the network.
AVGT	Average throughput per link over all links in the network (number of packets).
AVGU	Average link utilization over all links in the network.
AVGTPS	Average link throughput (packets/sec).
AVGD	Mean packet delay measured over all PS nodes in the network.
AVGB	Fraction of voice calls blocked measured over all CS nodes in the network.
IADD	Indicates whether a link will be added (1), deleted (-1), or neither added nor deleted (0). Used by INFACE to determine when new routing tables need to be generated.
IRSAVE	Indicates when a perturbation failure occurs as a result of a link deletion (2). Used by INFACE to determine when the routing tables need to be reconstructed.
COST	Cost ($) of the current topology. Determined in COSTOP.
IPASS	A counter to indicate how many passes (iterations) of the outer feedback loop mechanism have been performed.

AD-A141 309 ADAPTIVE TOPOLOGICAL CONFIGURATION OF AN INTEGRATED CIRCUIT/PACKET-SWITCHED COMPUTER NETWORK(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH M J KIEMELE 1984 3/3

UNCLASSIFIED AFIT/CI/NR-84-16D F/G 17/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

VARIABLE	Description
NSHUFS	Number of shuffles or randomizations to be performed on the original node locations. For NSHUFS > 1, the shuffle resulting in the lowest cost topology is chosen as the starting topology.
CAVAIL(MAXDEG)	An array storing the channels that are available as alternate paths eminating from a given node. The dimension need not be greater than (MAXDEG-2), where MAXDEG is the maximum degree of all the nodes.
NSLOTS(NCAPS)	Number of slots associated with each line type.
DIMLIM(3)	Model dimension limits for the total number of nodes, links, and slots.
FLAGD FLAGB FLAGR FLAGU	Flags that indicate (with a 1) when the delay, blocking, reliability, and utilization constraints, respectively, are satisfied.
FEAS	FEAS = 1 indicates a feasible solution. FEAS = 1 only if FLAGD = FLAGB = FLAGR = 1.
BCC(2, NSITES)	Contains the nodes in each of two biconnected components, as determined in BICON.
NBCC(2)	Contains the number of nodes in each of two biconnected components.
CUTSET(MAXL)	Stores the links that have been determined to be in the saturated cut. MAXL is the maximum number of links allowed.

VARIABLE	Description
COMP1(NSITES) COMP2(NSITES)	Contains the nodes in each of the two components which are separated by the cutset.
COMPOF(NSITES)	Identifies the component (1 or 2) that each node is in.
STDEVS(NSITES, 2)	Number of standard deviations that the performance measures mean packet delay (1) and blocking (2) at each node are away from their respective total system means.
FAILED(NSITES, NSITES)	Maintains a record of perturbations that have failed and that should not be attempted again.

APPENDIX C

Appendix C describes each of the major tables in the simulator. Each table contains a description of its composition and use. The array names and arguments associated with each table are given parenthetically.

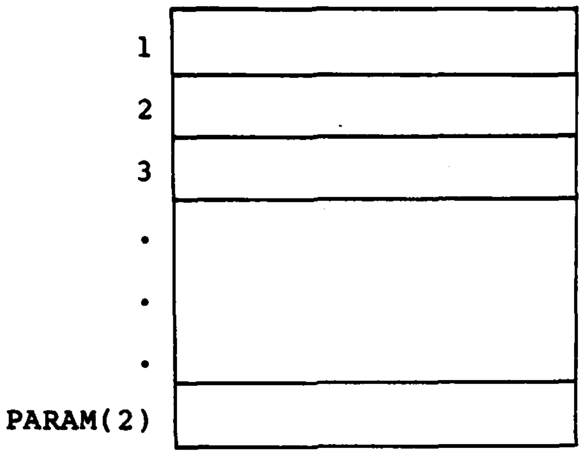
Table C1. Parameter table (PARAM)[X]

PARAM is a singly indexed array where X refers to the particular user input parameter. Table entries are:

1	Total number of PS and CS nodes
2	Total number of HDX channels
3	Total number of slots in the network
4	Ratio of packet to voice slots
5	Frame time duration
6	Fixed time routing delay per node
7	Circuit switch voice arrival rate
8	Packet switch transaction arrival rate
9	Starting time for simulation run
10	Ending time for simulation run
11	Packet switch saturation level
12	Voice digitization rate
13	Buffer size at each packet switch
14	Average voice call service time
15	Number of bits per packet
16	Average number of packets per message
17	System error run time

Table C2. Slot table (PARM3)[X]

This table contains the number of slots allocated to each HDX channel X, where X = 1, 2, 3,..., PARAM(2).



The relationship between PARM3 and PARAM(3) is

$$\text{PARAM}(3) = \sum_{i=1}^{\text{PARAM}(2)} \text{PARM3}(i).$$

Table C3. Event table (EVTBL)[Node, Entry]

This table maintains the next event occurrence at each node. The [Node, Entry] table entries are:

	1	2	3	4	5
1					
2					
3					
.					
.					
.					
PARAM(1)					

Entry	Definition
1	Time (msec)
2	Type 1, 2, 3, 4 1 = Class II (data) arrival 2 = Class II (data) departure 3 = Class I (voice) arrival 4 = Class I (voice) departure
3	Message length if Class II or time of departure if Class I
4	Final destination
5	Queue address (pointer into Queue table)

Table C4. Destination table (DESTAB)[Node, Dest]

This table gives the primary routing channel between each node pair [Node, Dest]. If the source node is a packet node, the [Node, Dest] entry contains the number of the directly connected circuit switch node instead of a channel number.

	1	2	3	. . . . .	PARAM(1)
1					
2					
3					
. . .					
PARAM(1)					

Table C5. Alternate destination table (DSTALT)
[Node, Dest]

Similar to Table C4, this table gives the alternate routing channel between each node pair [Node, Dest]. Like the primary destination table, each main diagonal element is zero.

	1	2	3	. . . . .	PARAM(1)
1					
2					
3					
. . .					
PARAM(1)					

Table C6. Channel table (CHANTB)[Channel, Entry]

Each row of the channel table corresponds to a particular slot in the network. For each slot, this table maintains the 11 attributes described below.

	1	2	3	4	5	6	7	8	9	10	11
1											
2											
3											
. . .											
PARAM(3)											

Entry	Definition
1	Final destination node for this transaction
2	Time the slot is active
3	Time the slot is available
4	Number of slots used for this transaction
5	Destination number of intermediate node
6	Cumulative time slot is in use
7	Source node for this transaction
8	Queue address (pointer into Queue table)
9	Cumulative number of transactions
10	Cumulative number of packets
11	Cumulative number of voice calls

Table C7. Queue table (QUEUE)[Node, Entry]

This table exists for packet switch nodes only. Each row, which corresponds to a packet switch node, maintains a record of all data transaction arrivals at that node. Each data transaction requires six entries in the table. These entries are described below. QUEUE is currently dimensioned at (26, 1800), thereby allowing 300 transactions at each packet switch node.

	1	2	3	4	5	6	. . . . . .	1800
1								
2								
3								
. . .								
PARAM(1)/2								

Entry	Definition
1	Priority
2	Transaction arrival time
3	Transaction departure time from the system
4	Total number of packets in the transaction
5	Final destination node
6	Number of messages in the transaction

Table C8. Call queue table (CALLQ)[Knode, Entry]

This table exists for circuit switch nodes only. Each row, which corresponds to a circuit switch node, maintains a record of all departing calls at that node. Each call requires the four entries described below. Current program dimensioning is CALLQ (26, 200), thereby allowing 50 calls at each circuit switch node.

	1	2	3	4	. . . .	200
1						
2						
3						
.						
.						
.						
PARAM(1)/2						

<u>Entry</u>	<u>Definition</u>
1	Call departure time from the system
2	Final destination node
3	Channel address pointer
4	Source or destination node, depending on which pass

Table C9. Calls accepted/rejected table
(CALLS)[Knode, Entry]

This table exists for circuit switch nodes only. Each row, which corresponds to a circuit switch node, maintains a cumulative record of the number of voice calls at that node that have been either accepted, rejected, or are still in the network.

	1	2	3
1			
2			
3			
.			
.			
.			
PARAM(1)/2			

Entry	Definition
1	Number of calls accepted
2	Number of calls rejected
3	Number of calls still in the system

Table C10. Circuit switch arrival table
(CSARV)[Knode, Entry]

This table exists for circuit switch nodes only. Each row contains information relating to the next voice call arrival at that node.

	1	2	3
1			
2			
3			
.			
.			
.			
PARAM(1)/2			

Entry	Definition
1	Time of arrival
2	Time of departure
3	Final destination node

Table C11. Queue entry count table (QCNT)[Node]

This table reflects the current number of transactions in each packet switch queue and the cumulative number of voice call departures for each circuit switch node.

1	Packet switch queue count
.	
.	
.	
PARAM(1)/2	Packet switch queue count
PARAM(1)/2 + 1	Cumulative voice call departs
.	
.	
.	
PARAM(1)	Cumulative voice call departs

Table C12. Cumulative time table (CUMTIM)[Node, Entry]

This table maintains a cumulative frequency distribution of packet delays at all packet switch nodes.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1													
2													
3													
.													
.													
.													
PARAM(1)/2													

Entry	Definition
1	$<$ .1 sec
2	$<$ .2 sec
3	$<$ .3 sec
4	$<$ .4 sec
5	$<$ .5 sec
6	$<$ .6 sec
7	$<$ .7 sec
8	$<$ .8 sec
9	$<$ .9 sec
10	$<$ 1 sec
11	$<$ 2 sec
12	$<$ 5 sec
13	$\geq$ 5 sec

Table C13. Seed table (SEEDTB)[Node, Entry]

This table contains the seeds used to generate arrival and departure times, message lengths, voice call service times, and destination numbers. The original seeds are read in as user specified seeds, but the simulator changes the seed table entry each time a seed is used.

	1	2	3	4
1				
2				
3				
.				
.				
.				
PARAM(1)				

Entry	Definition
1	Arrival times
2	Departure times
3	Geometric message lengths (Class II)
	Voice call service times (Class I)
4	Destination nodes

Table C14. Channel connectivity table (NODCHL)[Channel]

This table is a quasi-inverse of the routing tables. That is, for each HDX channel (row), the entry is the receiving or destination node for that channel.

1	Destination node (1)
2	Destination node (2)
3	Destination node (3)
.	
.	
.	
PARAM(2)	Destination node (PARAM(2))

SORCHL[Channel] is this table's counterpart. Its entry for each HDX channel is the source node from which the channel eminates.

Table C15. Alternate channel table (ALTCH)[Channel]

This table is a working table which is designed to reduce table look-up time. The entry for each channel (row) is a "0" or "1". A "1" indicates that this channel is being used as an alternate channel in the route under construction. A "0" indicates that it may be a primary channel or that it may not be under consideration at all.

1	0/1
2	0/1
3	0/1
.	
.	
.	
PARAM(2)	0/1

This table is always initialized to zeros prior to route construction.

Table C16. Link availability table (NLINES)[Channel]

Each independent HDX channel can be thought of as consisting of a number of lines or slots. This table is a working table that maintains a current count of the number of available slots for each channel.

1	PARM3 (1)
2	PARM3 (2)
3	PARM3 (3)
.	
.	
.	
PARAM(2)	PARM3 (PARAM(2))

Initially, all slots are available so the table originally appears with the PARM3 values as the entries, as shown here.

Table C17. Link table (LINKTB)[Node, Dest]

This is a dynamic working table containing a channel address pointer for each active node/destination connection. The entry for each node pair is a channel number and it is used to reduce table look-ups. Diagonal elements are not used and are set to zero.

	1	2	3	. . .	PARAM(1)
1					
2					
3					
. . .					
PARAM(1)					

APPENDIX D

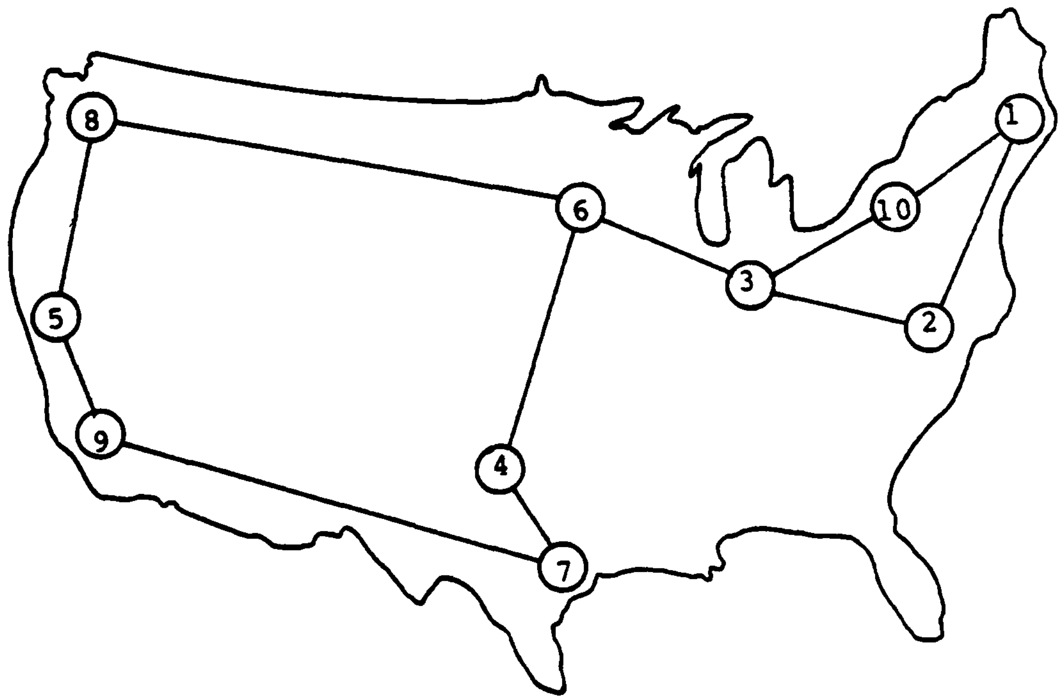
Appendix D provides sample input data required by the model. This data is the input data that was used for Case 3 in the application of the model to a 20-node integrated network (ref. Chapter VI).

```
  10
19.4        10.5        23.10       10.9        14.3        4.2
13.4        6.1         25.5        13.8        15.1        13.9
24.3        12.4        2.4         8.3         1.1         11.1
2.8         17.2
0.0   4.45  4.45  4.45  4.45  4.45  4.45  4.45  4.45  4.45
4.45  0.0   4.45  4.45  4.45  4.45  4.45  4.45  4.45  4.45
4.45  4.45  0.0   4.45  4.45  4.45  4.45  4.45  4.45  4.45
4.45  4.45  4.45  0.0   4.45  4.45  4.45  4.45  4.45  4.45
4.45  4.45  4.45  4.45  0.0   4.45  4.45  4.45  4.45  4.45
4.45  4.45  4.45  4.45  4.45  0.0   4.45  4.45  4.45  4.45
4.45  4.45  4.45  4.45  4.45  4.45  0.0   4.45  4.45  4.45
4.45  4.45  4.45  4.45  4.45  4.45  4.45  0.0   4.45  4.45
4.45  4.45  4.45  4.45  4.45  4.45  4.45  4.45  0.0   4.45
4.45  4.45  4.45  4.45  4.45  4.45  4.45  4.45  4.45  0.0
0.0   0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45
0.45  0.0   0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45
0.45  0.45  0.0   0.45  0.45  0.45  0.45  0.45  0.45  0.45
0.45  0.45  0.45  0.0   0.45  0.45  0.45  0.45  0.45  0.45
0.45  0.45  0.45  0.45  0.0   0.45  0.45  0.45  0.45  0.45
0.45  0.45  0.45  0.45  0.45  0.0   0.45  0.45  0.45  0.45
0.45  0.45  0.45  0.45  0.45  0.45  0.0   0.45  0.45  0.45
0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.0   0.45  0.45
0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.0   0.45
0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.0
180.  180.  180.  180.  180.  180.  180.  180.  180.  180.
   10 1000
  5
800000.     1000000.    1200000.    1600000.    2000000.
1.          2.          4.          8.          16.
50.         50.         100.        150.        200.
 2          .05         .02         .01
1.0
500.
.60
.10
 4747      0
06413 46427 86799 19565
17767 05431 35635 99817
26803 20505 14523 81949
49329 28573 16213 78317
15307 08391 00597 32537
45611 49883 09303 71715
36147 68607 98083 58401
64969 08015 79953 08721
89837 88159 25241 75379
10851 50949 06571 37143
17581 17153 49503  2081
48927 17347 17157 24195
74603 54027 56303 10187
39813 21305 69047  2775
10521 73373 49813 92539
84169 26867 44787 22419
78415 49233 77593 96597
16031 17995 60663 29709
72135 28107   377 24891
85717 50947 30947 86241
```

APPENDIX E

Appendix E provides sample output from the execution of the model. The data presented here is from Case 3 in the application of the model to a 20-node integrated network (ref. Chapter VI). An abbreviated output for the first three iterations of this case is given.

To facilitate the examination of the output, the following diagram is provided. The node numbers correspond to the randomized nodes, and the links shown are that of the starting topology. The exact capacities of each link can be obtained from the CONECT matrix. The output also uses circuit switch node numbers 11, 12,..., 20. These are assumed to be at the same locations as nodes 1, 2,..., 10, respectively.



INPUT DATA

NO. OF SITES: 10

SITE COORDINATES:

X(1)=	19.40	Y(1)=	10.50	X(2)=	23.10	Y(2)=	10.90
X(3)=	14.30	Y(3)=	4.20	X(4)=	13.40	Y(4)=	6.10
X(5)=	25.50	Y(5)=	13.80	X(6)=	15.10	Y(6)=	13.90
X(7)=	24.30	Y(7)=	12.40	X(8)=	2.40	Y(8)=	8.30
X(9)=	1.10	Y(9)=	11.10	X(10)=	2.80	Y(10)=	17.20

PACKET SWITCH TRAFFIC (MESSAGE ARRIVALS/SEC)

	1	2	3	4	5	6	7	8	9	10
1	0.0	4.45	4.45	4.45	4.45	4.45	4.45	4.45	4.45	4.45
2	4.45	0.0	4.45	4.45	4.45	4.45	4.45	4.45	4.45	4.45
3	4.45	4.45	0.0	4.45	4.45	4.45	4.45	4.45	4.45	4.45
4	4.45	4.45	4.45	0.0	4.45	4.45	4.45	4.45	4.45	4.45
5	4.45	4.45	4.45	4.45	0.0	4.45	4.45	4.45	4.45	4.45
6	4.45	4.45	4.45	4.45	4.45	0.0	4.45	4.45	4.45	4.45
7	4.45	4.45	4.45	4.45	4.45	4.45	0.0	4.45	4.45	4.45
8	4.45	4.45	4.45	4.45	4.45	4.45	4.45	0.0	4.45	4.45
9	4.45	4.45	4.45	4.45	4.45	4.45	4.45	4.45	0.0	4.45
10	4.45	4.45	4.45	4.45	4.45	4.45	4.45	4.45	4.45	0.0

CIRCUIT SWITCH TRAFFIC (VOICE ARRIVALS/MIN)

	1	2	3	4	5	6	7	8	9	10
1	0.0	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
2	0.45	0.0	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
3	0.45	0.45	0.0	0.45	0.45	0.45	0.45	0.45	0.45	0.45
4	0.45	0.45	0.45	0.0	0.45	0.45	0.45	0.45	0.45	0.45
5	0.45	0.45	0.45	0.45	0.0	0.45	0.45	0.45	0.45	0.45
6	0.45	0.45	0.45	0.45	0.45	0.0	0.45	0.45	0.45	0.45
7	0.45	0.45	0.45	0.45	0.45	0.45	0.0	0.45	0.45	0.45
8	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.0	0.45	0.45
9	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.0	0.45
10	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.0

CIRCUIT SWITCH SERVICE TIMES (SEC)

1	2	3	4	5	6	7	8	9	10
180.	180.	180.	180.	180.	180.	180.	180.	180.	180.

AVG. NO. OF PACKETS/MESSAGE = 10

NO. OF BITS/PACKET = 1000

LINE CAPACITY AND COST INFORMATION:

LINE TYPE	CAPACITY (BPS)	COST($) PER UNIT LENGTH	FIXED COST($)
1	800000.	1.00	50.00
2	1000000.	2.00	50.00
3	1200000.	4.00	100.00
4	1600000.	8.00	150.00
5	2000000.	16.00	200.00

RELIABILITY CONSTRAINTS:

NODE CONNECTIVITY (MIN. NO. OF NODES THAT MUST BE DELETED TO CAUSE A DISCONNECT)= 2

PROBABILITY OF LINK FAILURE=0.050

DISCONNECT PROBABILITY BOUND=0.020

BOUND ON FRACTION OF NODES UNABLE TO COMMUNICATE=0.010

DELAY CONSTRAINT:

MEAN PACKET DELAY SHOULD NOT EXCEED 1.000 SECONDS.

THROUGHPUT CONSTRAINT:

AVERAGE NUMBER OF PACKETS FLOWING THROUGH EACH CHANNEL SHOULD BE AT LEAST 500. PACKETS/SEC.

LINK UTILIZATION CONSTRAINT:

MINIMUM AVERAGE LINK UTILIZATION SHOULD BE 0.60

```
CS BLOCKING CONSTRAINT:
MAXIMUM ALLOWABLE AVERAGE SYSTEM BLOCKING IS 0.10


IPASS=    1****************
     ISEED=   4747
     NSHUFS=     1
     ISHUF=     1
     THE RANDOMIZED NODES:
          I        X(I)       Y(I)   MAP(I)
          1       25.50      13.80    5
          2       23.10      10.90    2
          3       19.40      10.50    1
          4       13.40       6.10    4
          5        1.10      11.10    9
          6       15.10      13.90    6
          7       14.30       4.20    3
          8        2.80      17.20   10
          9        2.40       8.30    8
         10       24.30      12.40    7
     THE CONECT MATRIX NOW LOOKS LIKE:
          1  2  3  4  5  6  7  8  9 10
      1   0  1  0  0  0  0  0  0  0  4
      2   1  0  2  0  0  0  0  0  0  0
      3   0  2  0  0  0  5  0  0  0  5
      4   0  0  0  0  0  4  3  0  0  0
      5   0  0  0  0  0  0  0  5  4  0
      6   0  0  5  4  0  0  0  5  0  0
      7   0  0  0  3  0  0  0  0  1  0
      8   0  0  0  0  5  5  0  0  0  0
      9   0  0  0  0  4  0  1  0  0  0
     10   4  0  5  0  0  0  0  0  0  0
     THE COST($) OF THIS TOPOLOGY IS:    2112.39

                                        SYSTEM PARAMETERS
NODES LINKS SLOTS RATIO SLOT NODE  CS       PS  MSG START TIME END TIME
                        TIME DELAY ARRIVAL ARRIVAL
   20    22   996    3   10MS 50 MS   4MIN   40SEC        0 MS  600000MS

    SEED TABLES:
          6413          46427          86799          19565
         17767           5431          35635          99817
         26803          20505          14523          81949
         49329          28573          16213          78317
         15307           8391            597          32537
         45611          49883           9303          71715
         36147          68607          98083          58401
         64969           8015          79953           8721
         89837          88159          25241          75379
         10851          50949           6571          37143
         17581          17153          49503           2081
         48927          17347          17157          24195
         74603          54027          56303          10187
         39813          21305          69047           2775
         10521          73373          49813          92539
         84169          26867          44787          22419
         78415          49233          77593          96597
         16031          17995          60663          29709
         72135          28107            377          24891
         85717          50947          30947          86241

     ROUTING TABLES ARE UPDATED NOW.
     PRIMARY ROUTING TABLE:
  0  1  3  3  3  3  3  3  3  3
  2  0  5  5  5  5  5  5  5  2
  9  6  0  7  7  7  7  7  7  9
 11 11 11  0 13 11 13 11 13 11
```

15	15	15	17	0	15	17	15	17	15
8	8	8	12	19	0	12	19	19	8
14	14	14	14	21	14	0	21	21	14
20	20	20	20	16	20	16	0	16	20
18	18	18	22	18	18	22	18	0	18
4	4	10	10	10	10	10	10	10	0

ALTERNATE ROUTING TABLE:

0	3	1	1	1	1	1	1	1	1
5	0	2	2	2	2	2	2	2	5
6	7	0	6	6	6	6	6	6	6
13	13	13	0	11	13	11	13	11	13
17	17	17	15	0	17	15	17	15	17
12	12	12	8	8	0	8	8	12	12
21	21	21	21	14	21	0	14	14	21
16	16	16	16	20	16	20	0	20	16
22	22	22	18	22	22	18	22	0	22
10	10	4	4	4	4	4	4	4	0

SOURCE, DESTINATION, AND CAPACITY (IN SLOTS) FOR EACH CHANNEL:

CHAN	SORCHL(I)	NODCHL(I)	PARM3(I)
1	11	12	24
2	12	11	24
3	11	20	48
4	20	11	48
5	12	13	30
6	13	12	30
7	13	16	60
8	16	13	60
9	13	20	60
10	20	13	60
11	14	16	48
12	16	14	48
13	14	17	36
14	17	14	36
15	15	18	60
16	18	15	60
17	15	19	48
18	19	15	48
19	16	18	60
20	18	16	60
21	17	19	24
22	19	17	24

SYSTEM PERFORMANCE MEASURES FOR THE GIVEN INPUT PARAMETERS

NODES	LINKS	SLOTS	RATIO	SLOT TIME	NODE DELAY	CS ARRIVAL	PS ARRIVAL	MSG START TIME	END TIME
20	22	996	3	10MS	50 MS	4MIN	40SEC	0 MS	600027MS

CHAN	THROUGHPUT	UTILIZATION
1	129819	0.326
2	94572	0.326
3	529641	0.549
4	608463	0.549
5	400518	0.550
6	459081	0.550
7	1365507	0.939
8	1622454	0.939
9	904221	0.604
10	815100	0.604
11	754473	0.631
12	703377	0.631
13	520986	0.577
14	567561	0.577
15	889008	0.625
16	871245	0.625
17	684390	0.680
18	672432	0.680

19	1002795	0.746
20	1136283	0.746
21	379263	0.769
22	384888	0.769

AVG NO OF PACKETS PER LINK= 704367
AVG LINK UTILIZATION = 0.636
AVG LINK THROUGHPUT (PACKETS/SEC)= 1174

PACKET NODE SUMMARY

NODE	AVG PACKET DELAY (SEC)	DATA TRANSACTIONS IN SYSTEM
1	20.710	32
2	19.470	18
3	21.974	39
4	8.614	30
5	12.477	43
6	10.283	37
7	21.170	53
8	10.329	36
9	11.860	38
10	13.571	15

AVG PACKET DELAY (SEC)= 14.937
AVG NO OF DATA TRANSACTIONS AT A NODE= 34.1

CS NODE SUMMARY

NODE	TOTAL CALLS	CALLS LOST	BLOCKING	CALLS IN SYSTEM
11	35	14	0.400	7
12	37	20	0.541	4
13	35	12	0.343	7
14	41	12	0.293	6
15	35	10	0.286	8
16	40	13	0.325	7
17	45	22	0.489	7
18	34	10	0.294	4
19	35	12	0.343	5
20	38	14	0.368	2

AVG NO OF CALLS PER NODE= 37.5
FRACTION OF CALLS BLOCKED= 0.371
AVG NO OF CALLS IN SYSTEM PER NODE= 5.7
CLASS 2 (DATA) ARRIVALS = 4927
CALSS 2 (DATA) DEPARTS = 4634
CLASS 1 (CS) ARRIVALS = 375
CLASS 1 (CS) DEPARTS = 179

FLAGD=0 FLAGB=0 FLAGU=1 FLAGR=0 FEAS=0
COMP1= 4 5 6 7 8 9
COMP2= 1 2 3 10
BCC # = 1 CONTAINS THE FOLLOWING NODES:
8 5 9 7 4
WHILE THE ARTICULATION POINT 6 IS NOT INCLUDED IN THE BCC.
BCC # = 2 CONTAINS THE FOLLOWING NODES:
6
WHILE THE ARTICULATION POINT 3 IS NOT INCLUDED IN THE BCC.
BCC # = 2 CONTAINS THE FOLLOWING NODES:
10 3 2
WHILE THE ARTICULATION POINT 1 IS NOT INCLUDED IN THE BCC.
A NEW LINK WILL BE ADDED BETWEEN NODES 7 AND 2

IPASS= 2***************
THE CONECT MATRIX NOW LOOKS LIKE:

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	0	0	0	0	4
2	1	0	2	0	0	0	3	0	0	0
3	0	2	0	0	0	5	0	0	0	5
4	0	0	0	0	0	5	3	0	0	0
5	0	0	0	0	0	0	0	5	5	0
6	0	0	5	5	0	0	0	5	0	0
7	0	3	0	3	0	0	0	0	3	0
8	0	0	0	0	5	5	0	0	0	0
9	0	0	0	0	5	0	3	0	0	0

10 4 0 5 0 0 0 0 0 0 0
THE COST($) OF THIS TOPOLOGY IS: 2532.95

SYSTEM PARAMETERS

NODES	LINKS	SLOTS	RATIO	SLOT TIME	NODE DELAY	CS ARRIVAL	PS ARRIVAL	MSG START TIME	END TIME
20	24	1140	3	10MS	50 MS	4MIN	40SEC	0 MS	600000MS

SOURCE, DESTINATION, AND CAPACITY (IN SLOTS) FOR EACH CHANNEL:

CHAN	SORCHL(I)	NODCHL(I)	PARM3(I)
1	11	12	24
2	12	11	24
3	11	20	48
4	20	11	48
5	12	13	30
6	13	12	30
7	12	17	36
8	17	12	36
9	13	16	60
10	16	13	60
11	13	20	60
12	20	13	60
13	14	16	60
14	16	14	60
15	14	17	36
16	17	14	36
17	15	18	60
18	18	15	60
19	15	19	60
20	19	15	60
21	16	18	60
22	18	16	60
23	17	19	36
24	19	17	36

SYSTEM PERFORMANCE MEASURES FOR THE GIVEN INPUT PARAMETERS

NODES	LINKS	SLOTS	RATIO	SLOT TIME	NODE DELAY	CS ARRIVAL	PS ARRIVAL	MSG START TIME	END TIME
20	24	1140	3	10MS	50 MS	4MIN	40SEC	0 MS	600030MS

CHAN	THROUGHPUT	UTILIZATION
1	490287	0.908
2	361236	0.885
3	557739	0.524
4	701379	0.535
5	446157	0.703
6	492570	0.654
7	847641	0.876
8	694698	0.900
9	1074075	0.846
10	1116897	0.831
11	800649	0.544
12	748320	0.535
13	590571	0.448
14	512718	0.438
15	503598	0.632
16	558780	0.647
17	546183	0.449
18	655071	0.473
19	667575	0.520
20	579300	0.496
21	886926	0.672
22	827640	0.648
23	783738	0.874
24	656511	0.914

AVG NO OF PACKETS PER LINK= 670844
AVG LINK UTILIZATION = 0.665

AVG LINK THROUGHPUT (PACKETS/SEC)= 1118

PACKET NODE SUMMARY

NODE	AVG PACKET DELAY (SEC)	DATA TRANSACTIONS IN SYSTEM
1	10.933	37
2	12.100	47
3	2.242	41
4	3.413	48
5	7.124	34
6	1.499	26
7	8.266	47
8	1.698	38
9	12.956	71
10	6.837	40

AVG PACKET DELAY (SEC)= 6.569
AVG NO OF DATA TRANSACTIONS AT A NODE= 42.9

CS NODE SUMMARY

NODE	TOTAL CALLS	CALLS LOST	BLOCKING	CALLS IN SYSTEM
11	35	7	0.200	7
12	37	13	0.351	6
13	35	6	0.171	8
14	41	8	0.195	7
15	35	8	0.229	9
16	40	4	0.100	12
17	45	19	0.422	8
18	34	3	0.088	5
19	35	13	0.371	5
20	38	9	0.237	4

AVG NO OF CALLS PER NODE= 37.5
FRACTION OF CALLS BLOCKED= 0.240
AVG NO OF CALLS IN SYSTEM PER NODE= 7.1
CLASS 2 (DATA) ARRIVALS = 5859
CALSS 2 (DATA) DEPARTS = 5477
CLASS 1 (CS) ARRIVALS = 375
CLASS 1 (CS) DEPARTS = 214

FLAGD=0 FLAGB=0 FLAGU=1 FLAGR=1 FEAS=0
COMP1= 4 5 6 7 8 9
COMP2= 1 2 3 10
A NEW LINK WILL BE ADDED BETWEEN NODES 9 AND 2

IPASS= 3***************

THE CONECT MATRIX NOW LOOKS LIKE:

	1	2	3	4	5	6	7	8	9	10
1	0	4	0	0	0	0	0	0	0	4
2	4	0	3	0	0	0	5	0	3	0
3	0	3	0	0	0	5	0	0	0	5
4	0	0	0	0	0	5	4	0	0	0
5	0	0	0	0	0	0	0	5	5	0
6	0	0	5	5	0	0	0	5	0	0
7	0	5	0	4	0	0	0	0	5	0
8	0	0	0	0	5	5	0	0	0	0
9	0	3	0	0	5	0	5	0	0	0
10	4	0	5	0	0	0	0	0	0	0

THE COST($) OF THIS TOPOLOGY IS: 3442.37

SYSTEM PARAMETERS

NODES	LINKS	SLOTS	RATIO	SLOT TIME	NODE DELAY	CS ARRIVAL	PS ARRIVAL	MSG START TIME	END TIME
20	26	1392	3	10MS	50 MS	4MIN	40SEC	0 MS	600000MS

SOURCE, DESTINATION, AND CAPACITY (IN SLOTS) FOR EACH CHANNEL:

CHAN	SORCHL(I)	NODCHL(I)	PARM3(I)
1	11	12	48
2	12	11	48
3	11	20	48
4	20	11	48

5	12	13	36
6	13	12	36
7	12	17	60
8	17	12	60
9	12	19	36
10	19	12	36
11	13	16	60
12	16	13	60
13	13	20	60
14	20	13	60
15	14	16	60
16	16	14	60
17	14	17	48
18	17	14	48
19	15	18	60
20	18	15	60
21	15	19	60
22	19	15	60
23	16	18	60
24	18	16	60
25	17	19	60
26	19	17	60

SYSTEM PERFORMANCE MEASURES FOR THE GIVEN INPUT PARAMETERS

NODES	LINKS	SLOTS	RATIO	SLOT TIME	NODE DELAY	CS ARRIVAL	PS ARRIVAL	MSG START TIME	END TIME
20	26	1392	3	10MS	50 MS	4MIN	40SEC	0 MS	600090MS

CHAN	THROUGHPUT	UTILIZATION
1	748398	0.767
2	704508	0.768
3	652074	0.610
4	710433	0.610
5	429969	0.533
6	426594	0.525
7	564060	0.436
8	559158	0.441
9	533349	0.708
10	535356	0.708
11	929607	0.731
12	906363	0.726
13	588351	0.420
14	572835	0.420
15	467445	0.366
16	462663	0.371
17	515508	0.530
18	537882	0.524
19	493572	0.408
20	518706	0.408
21	764217	0.638
22	791337	0.638
23	748824	0.597
24	746598	0.597
25	497382	0.389
26	489072	0.389

AVG NO OF PACKETS PER LINK= 611317
AVG LINK UTILIZATION = 0.548
AVG LINK THROUGHPUT (PACKETS/SEC)= 1019

PACKET NODE SUMMARY

NODE	AVG PACKET DELAY (SEC)	DATA TRANSACTIONS IN SYSTEM
1	0.429	31
2	0.234	28
3	0.203	26
4	0.201	14
5	0.264	24
6	0.167	12
7	0.218	19

8	0.212	19
9	0.238	18
10	0.293	42

AVG PACKET DELAY (SEC)= 0.245
AVG NO OF DATA TRANSACTIONS AT A NODE= 23.3

CS NODE SUMMARY

NODE	TOTAL CALLS	CALLS LOST	BLOCKING	CALLS IN SYSTEM
11	35	3	0.086	10
12	37	1	0.027	10
13	35	0	0.0	9
14	41	1	0.024	9
15	35	1	0.029	11
16	40	0	0.0	13
17	45	0	0.0	16
18	34	2	0.059	5
19	35	0	0.0	6
20	38	4	0.105	4

AVG NO OF CALLS PER NODE= 37.5
FRACTION OF CALLS BLOCKED= 0.032
AVG NO OF CALLS IN SYSTEM PER NODE= 9.3
CLASS 2 (DATA) ARRIVALS = 6052
CALSS 2 (DATA) DEPARTS = 5872
CLASS 1 (CS) ARRIVALS = 375
CLASS 1 (CS) DEPARTS = 270
FLAGD=1 FLAGB=1 FLAGU=0 FLAGR=1 FEAS=1
A FEASIBLE SOLUTION HAS BEEN FOUND.
STORE IT AND MAKE PERTURBATIONS FROM THIS TOPOLOGY.
LINK # 8 WHICH CONNECTS NODES 4 AND 6 IS REDUCED BY 3 UNITS.

VITA

Mark Jay Kiemele was born on December 12, 1947, in Bismarck, North Dakota. He graduated from Linton High School, Linton, North Dakota, in 1965, and earned his Bachelor of Science and Master of Science degrees in Mathematics from North Dakota State University in 1969 and 1970, respectively.

Mr. Kiemele received his Air Force commission via R.O.T.C. in 1969 and entered active duty in May, 1971. His assignments have been as an aircraft survivability/vulnerability analyst (1971-74), a scientist exchange officer to the Federal Republic of Germany (1974-76), and a computer software engineer on the development of the Cruise Missile weapon system (1976-79). Most recently he was assigned as an Instructor/Assistant Professor, Department of Mathematical Sciences, U. S. Air Force Academy, Colorado (1979-81). Major Kiemele has completed Squadron Officer School and Air Force Air Command and Staff College.

Mr. Kiemele married Carol Mary Jahr on September 11, 1971, and they have a son, Kyle, age 6. His permanent mailing address is:

617 1st St. NE
Linton, North Dakota 58552

The typist for this dissertation was Debbie Callaway.

END

FILMED

DTIC